Franzis
Computer-Praxis

Klein

Basic-Interpreter

Funktionsweise und Implementierung in 8080/Z-80-Computern

```
02B8                    PSEUDO:
02B8   57F0 1E24        POP INSTRW,R15^:
02BC   57F0 1E20        POP FCWSTO,R15^:
02C0   57F0 1E22        POP PCSTO,R15^:
02C4   7603 0362        LD R3,^TXTPS:
02C8   D036             CALR PRINT11:
02CA   6101 1E24        LD R1,INSTRW:
02CE   D049             CALR PRR1SP:
02D0   6101 1E22        LD R1,PCSTO:
02D4   AB11             DEC R1,2: %GENAUER STAND
02D6   5F00 023E        CALL PRR1SP:
02DA   D073             CALR CRLF111:
02DC   5E08 0052        JP LOOPMAIN:
02E0                    SYSCALL:
02E0   5C09 000F 1E00   LDM REGSTO,R0,16: %SAVE REGS
02E6   97F2             POP R2,R15^:     %IDENT
02E8   0702 00FF        AND R2,0FFH:
02EC   57F0 1E20        POP FCWSTO,R15^:%DIREKT
02F0   57F0 1E22        POP PCSTO,R15^: %ADR NORM/SYS
02F4   0B02 0001        CP R2,1:        %CI NACH R1
02F8   5E0E 0300        JP NZ,SK1:
02FC   DFC1             CALR CI:
02FE   E820             JR FINA:
0300                    SK1:
0300   0B02 0002        CP R2,2:        %CO VON R1
0304   EE02             JR NZ,SK2:
0306   DFBC             CALR CO:
```

Mikrocomputer und Basic
Der Interpreter
Der Compiler
RDK-Basic
I/O-Routinen
„Tiny Basic“ und dessen Einsatz
Analyse von arithmetischen Ausdrücken
Befehlsverarbeitung
Basic-Programmbeispiele
Erweiterungsmöglichkeiten des Basic-Interpreters
Floppy-Anschluß
12-KByte für den Z80
Anpassung des Basic-Interpreters
12-KByte-Basic
Befehlsbeschreibung
Programmbeispiel ELIZA
Z8000-Basic
Z8000-Monitor

Franzis'

Klein, Basic-Interpreter

In der Reihe
Franzis Computer-Praxis
sind erschienen:

Klein, Mikrocomputersysteme
Klein, Mikrocomputer Hard- und Softwarepraxis
Klein, Z-80-Applikationsbuch
Plate, Pascal: Einführung – Programmentwicklung – Strukturen

Franzis Computer-Praxis

Rolf-Dieter Klein

Basic-Interpreter

Funktionsweise und Implementierung
in 8080/Z-80-Computern

Mit 43 Abbildungen

Franzis'

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Klein, Rolf-Dieter:
Basic-Interpreter: Funktionsweise u. Implementierung in 8080/Z-80-Computern / Rolf-Dieter Klein. – München: Franzis, 1981.
(Franzis-Computer-Praxis)
ISBN 3-7723-6941-3

Satz: SatzStudio Pfeifer, Germering
Druck: Franzis-Druck GmbH, Karlstraße 35, 8000 München 2
Printed in Germany · Imprimé en Allemagne

ISBN 3-7723-6941-3

Vorwort

Basic ist als Programmiersprache wohl jedermann geläufig, der sich mit Mikrocomputern beschäftigt. Schon bisher wurde viel über Basic geschrieben – die Art der Programmierung, die verschiedenen Dialekte, aber kaum etwas über die Realisierung von Basic-Interpretern. Hier soll nun eine Möglichkeit gegeben werden, hinter die Kulissen zu sehen und nicht nur in der Lage zu sein, in Basic programmieren zu können, sondern auch einen Basic-Interpreter selbst zu schreiben oder den hier beschriebenen Interpreter zu erweitern, umzubauen oder eigene Ideen hinzufügen zu können. Durch das bessere Verständnis der Wirkungsweise eines Interpreters wird der Leser auch in der Lage sein, Basic-Programme wirkungsvoller zu schreiben und manche Eigenheiten besser zu verstehen.

Das Buch gliedert sich im Prinzip in vier Abschnitte: Nach einer prinzipiellen Einführung über die unterschiedliche Arbeitsweise von Interpretern und Compiler wird das „RDK Basic" anhand eines Listings und Syntaxdiagramme ausführlich behandelt. Dann folgt ein Abschnitt, der insbesondere den Z80-Besitzern gilt: ein 12-K Byte-Basic mit sehr leistungsfähigem Befehlssatz, und schließlich für Z8000-Besitzer das RDK-Basic als Z8000-Variante, wobei hingegen der Rest des Buchs überwiegend in 8080-Maschinensprache geschrieben ist.

Auch für Leser, die keinen der genannten Prozessoren besitzen, ist es leicht, die Programme zu verstehen und ggf. für den eigenen Prozessor aufzubereiten.

Aber auch Leser, die sich „nur" in die Assemblersprache einarbeiten wollen, werden in den Listings viele Tricks und Kniffe kennenlernen und Einblick in die Programmierung gewinnen.

Rolf-Dieter Klein

Wichtiger Hinweis

Die in diesem Buch wiedergegebenen Schaltungen und Verfahren werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen und technischen Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag sieht sich deshalb gezwungen, darauf hinzuweisen, daß er weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen kann. Für die Mitteilung eventueller Fehler sind Autor und Verlag jederzeit dankbar.

Inhalt

1	**Mikrocomputer und Basic**	9
2	**Realisierungsmöglichkeiten von Basic**	10
2.1	Der Interpreter	10
2.2	Der Compiler	19
2.3	Teilübersetzung als Kompromiß	29
3	**RDK-Basic**	31
3.1	Ein „Tiny Basic" und dessen Einsatz	31
3.2	Hardware bei Z80 und Z8000 zum Betrieb des Tiny Basic	31
3.3	Realisierung des Basic-Interpreters	32
3.3.1	I/O-Routinen	66
3.3.2	Einfache zeichenverarbeitende Routinen	67
3.3.3	Analyse von arithmetischen Ausdrücken	70
3.3.4	Befehlsabarbeitung	77
3.3.5	Basic-Programmbeispiele	89
3.3.6	Erweiterungsmöglichkeiten des Basic-Interpreters	93
3.3.6.1	Fließkomma-Arithmetik	93
3.3.6.2	Floppy-Anschluß	101
3.3.6.3	Graphik-Erweiterungen	101
4	**12-KByte-Basic für Z80**	103
4.1	Anpassung des Basic-Interpreters	115
4.2	12-KByte-Basic-Befehlsbeschreibung	116
4.3	Programmbeispiel ELIZA	125
5	**Anhang**	129
5.1	Z8000-Basic	129
5.1.1	Z8000-Monitor	163
5.1.2	Der Basic-Interpreter	164
5.2	Literaturverzeichnis	164
5.3	Glossar	165
Sachverzeichnis		177

1 Mikrocomputer und Basic

B.A.S.I.C. (Beginners All Purpose Symbolic Instruction Code) wurde 1962 von John Kemeny und Thomas Kurtz „erfunden". Wegen der leichten Erlernbarkeit und der einfachen Realisierung wurde es sehr schnell bei Mikrocomputern als Programmiersprache eingesetzt und ist jetzt in einer Vielzahl von Dialekten und Erweiterungen auf dem Markt.

Gleich zu Beginn der Mikrocomputergeschichte kamen Basic-Interpreter auf. Sie waren zunächst sehr einfach und mit vielen Einschränkungen versehen. Basic wurde dann immer weiter ausgebaut, und es kamen sogar spezielle Basic-Rechner wie Apple, PET, CBM, TRS-80 usw. heraus. Bei Mikrocomputer-Neuentwicklungen gibt es nun immer schon schnell einen dazugehörigen Basic-Interpreter. Basic besitzt neben dem großen Vorteil der leichten Erlernbarkeit auch den Vorteil, daß Programmtests schnell durchgeführt werden können und mit wenig Aufwand ein lauffähiges Programm erstellt werden kann.

In den folgenden Kapiteln wird gezeigt, wie es möglich ist, einen Basic-Interpreter zu konstruieren.

2 Realisierungsmöglichkeiten von Basic

Der Mikrocomputer hat die Aufgabe, die eingegebenen Basic-Kommandos und Programme zu verstehen und auszuführen. Dafür gibt es im Prinzip drei Möglichkeiten:

Im ersten Fall versteht der Computer die Basic-Befehle unmittelbar und führt sofort, nachdem er einen Befehl verstanden hat, diesen aus. Dies nennt man Interpretation. Das dazugehörige Maschinenprogramm wird Interpreter genannt.

Dann gibt es noch die Möglichkeit die Basic-Befehle in ein Maschinenprogramm umzuwandeln und dann auszuführen. Es wird dazu ein Compiler benötigt. Die dritte Art stellt gewissermaßen eine Mischung aus Interpreter und Compiler dar, die Basic-Befehle werden in meist ein Byte lange Abkürzungen („Tokens", Intern-Darstellung) umgewandelt, und diese werden nach „RUN" interpretiert. Im folgenden sollen diese drei Arten anhand von kurzen Beispielen weiter verdeutlicht werden.

2.1 Der Interpreter

Ein Basic-Befehl wird erkannt und dann sofort ausgeführt. Um die Arbeitsweise eines Interpreters besser verstehen zu können, soll eine kleine Programmiersprache erfunden werden, für die ein Interpreter gebaut wird.

Die Programmiersprache besteht aus einbuchstabigen Befehlen, die von einem weiteren Byte, dem Operanden gefolgt sind. Sie ähnelt damit zwar mehr einer Assemblersprache für einen Prozessor, als einer höheren Programmiersprache, doch sonst wären wir ja schon bei dem „RDK-Basic", und es soll hier wirklich nur das Prinzip verdeutlicht werden.

Die Befehle lauten im einzelnen:

LOAD (adr)	La	Laden des auf der Adresse „adr" stehenden Wertes in den Akkumulator.
STORE adr	Sa	Der Inhalt des Akkumulators wird auf die Adresse „adr" gespeichert.
ADD (adr)	Aa	Zum Inhalt des Akkumulators wird der Inhalt des auf der Adresse „adr" stehenden Wertes addiert. Das Ergebnis steht also im Akku.
NEG	Nd	Der Inhalt des Akkumulators wird bitweise komplementiert. „d" steht für „dummy" und besitzt keine Bedeutung, muß allerdings vorhanden sein.
JUMP adr	Ja	Sprung auf die Adresse „adr" im Programmteil, dort wird der nächste Befehl ausgeführt.
JMPNZ adr	Ia	Falls der Inhalt des Akkumulators ungleich Null ist, wird auf die Adresse „adr" im Programmteil gesprungen.

OUTHEX (adr)	Oa	Der Inhalt der Speicherzelle auf „adr" in sedezimaler (hexadezimaler) Form auf die Konsole ausgegeben.
GETHEX (adr)	Ga	Von der Konsole wird ein sedezimaler (hexadezimaler) Wert geholt und auf die Speicherzelle „adr" geschrieben.
CRLF	Cd	Es wird ein Carriage Return Line Feed (CRLF, Wagenrücklauf/Zeilenvorschub) an die Konsole ausgegeben. „d" steht wieder für „dummy", da bei CRLF keine zusätzlichen Parameter nötig sind.
ZEICH (adr)	Za	Der Inhalt der Speicherzelle „adr" wird als ASCII-Zeichen interpretiert und dann auf der Konsole ausgegeben.
HALT	Hd	Das Programm ist beendet und es wird ein Stop ausgeführt. „d" steht für „dummy", da kein weiterer Parameter nötig ist.

Die hypothetische Maschine mit dem obigen Befehlssatz besitzt einen Akkumulator und kann 256 Bytes Programm und 256 Bytes Daten adressieren.

Ein Programm wird durch Hintereinanderschreiben der obigen Befehle gebildet. Dabei werden Parameter in binärer Form direkt hinter das Befehlswort geschrieben.

Eine Besonderheit ist, daß Daten adreßmäßig vom Programm völlig getrennt sind und Programmteile auch vom Programm her nicht erreichbar sind. Doch das ist ähnlich zu Basic, denn dort kann im Normalfall auch nicht vom Programm aus auf das Programm selbst zugegriffen werden. Ein Programm in unserer Programmsprache sieht z.B. wie folgt aus:

Gɸɸ Gɸ1 Lɸɸ Aɸ1 Sɸɸ Oɸɸ Hɸɸ
(Parameter in Sedezimal)

Ein Speicherauszug (sedezimal) sieht wie folgt aus:

47 ɸɸ 47 ɸ1 4C ɸɸ 41 ɸ1 53 ɸɸ 4F ɸɸ 48 ɸɸ

Das Programm bewirkt das Einlesen zweier Zahlen, addiert diese, dann wird die Summe ausgegeben und das Programm mit einem Halt beendet.

Abb. 2.1-1 zeigt die Hauptschleife des Interpreterprogramms. Als erstes wird der Programmzähler mit einem Startwert vorbesetzt. Von dieser Adresse wird nach dem erstmaligen Start des Interpreters der erste Befehl geholt.

Nun beginnt die eigentliche Schleife. Es wird der Befehl vom Speicher geholt. Dabei wird hier die Befehlskurzbezeichnung sowie der darauf folgende Datenwert geladen. Daher ist es auch nötig, im Programm immer einen Parameter mit anzugeben, auch wenn der Befehl selbst ihn gar nicht benötigt, zum Beispiel beim HALT-Befehl. Der Programmzähler wird dann erhöht – in unserem Fall um den Wert zwei, da ein Befehl zwei Speicheradressen belegt. Es folgt nun die Abfrage auf die einzelnen Befehle. Trifft einer der Fälle zu, so wird das dazugehörige Teilprogramm angesprungen. Wurde kein Befehl erkannt, so wird auf eine Fehlerroutine gesprungen.

Abb. 2.1-2 zeigt die einzelnen Teilprogramme. Bei LOAD zum Beispiel wird der Parameter des Befehls als Adresse interpretiert und der Inhalt dieser Speicherzelle wird in den Akkumulator geladen. Beim

Abb. 2.1-1 Hauptschleife des Interpreters

START

Programm-Adresse Laden → PC

LOOP

Befehl Laden
PC + 1 → PC

LOAD J → Lade Befehl
N
STORE J → Speichere Befehl
N
ADD J → Addieren
N
NEG J → Negieren
N
JUMP J → Sprung unbedingt
N
JMPNZ J → Sprung, wenn nicht 0
N
OUTHEX J → Hex Ausgabe
N
GETHEX J → Hex Eingabe
N
CRLF J → Carriage return
N
ZEICH J → Zeichenausgabe
N
HALT J → Halt der CPU
N
GOTO FEHLER

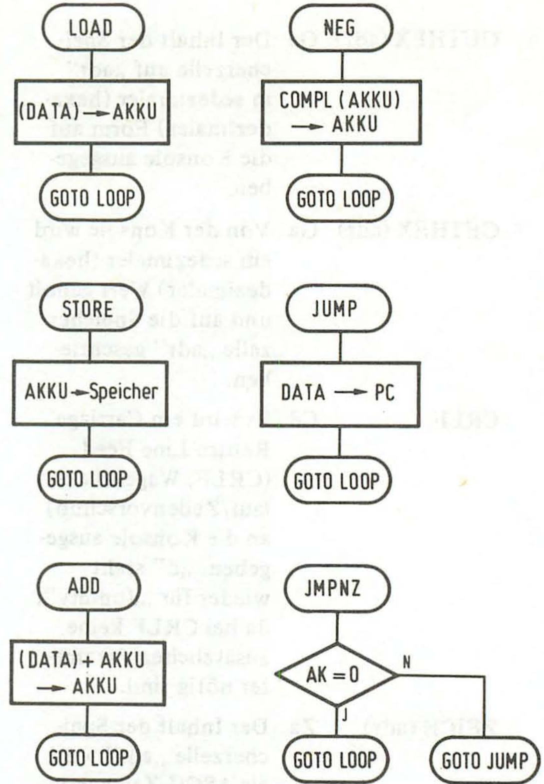
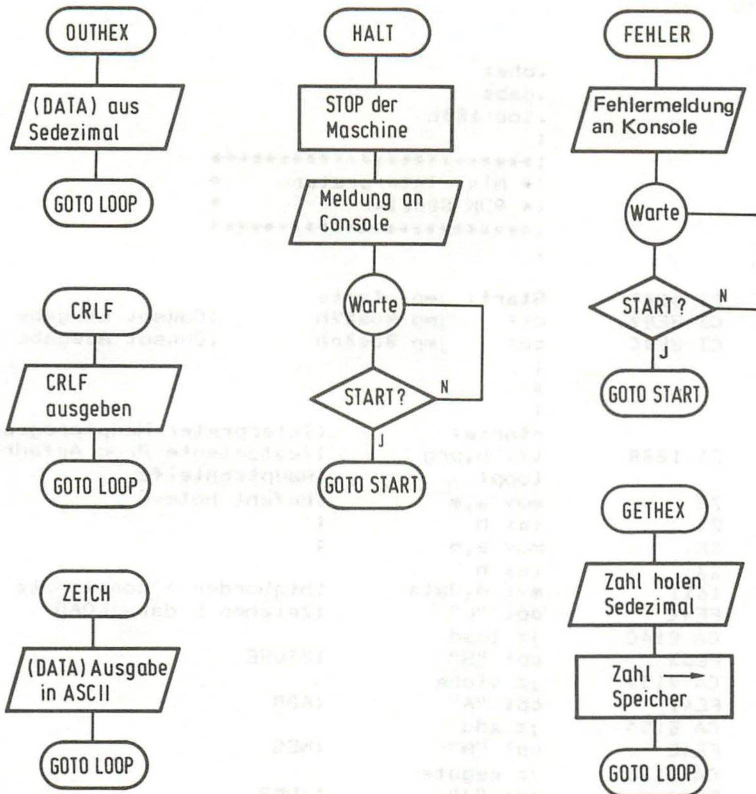


Abb. 2.1-2 Teilprogramme des Interpreters

STORE-Befehl ist es entsprechend umgekehrt. Beim Sprungbefehl JUMP wird der Parameter direkt in den Programmzähler gespeichert und damit der Sprung ausgeführt. Beim bedingten Sprung wird zuvor noch der Akkumulatorinhalt auf Null getestet und der Sprung ausgeführt falls der Inhalt ungleich Null ist.

Beim HALT-Befehl wird die Programmausführung gestoppt und eine Meldung auf der Konsole ausgegeben. Es wird anschließend gewartet bis eine Taste betätigt wurde und dann der Interpreter von neuem gestartet (Rücksetzstart). Tritt ein Fehler auf, so geschieht genau das Gleiche. Es wird nur eine andere Meldung ausgegeben.



zu Abb. 2.1-2

Abb. 2.1-3 zeigt das Listing des Programms. Das Programm beginnt mit drei Sprüngen. Der erste Sprung führt auf den eigentlichen Programmstart mit der Marke STARTE. Dann folgen zwei Sprünge mit absoluten Sprungzielen. Der erste Sprung mit der Marke CI führt auf eine Konsoleingaberoutine und es wird ein Zeichen von der Tastatur der Konsole im Akkumulator übergeben. Der Sprung CO enthält im C-Register ein Zeichen, welches vom Unterprogramm auf der Adresse BE0CH an die Konsole ausgegeben wird. Die Unterprogramme enden mit einem RET-Befehl, obwohl es auf den ersten Blick sehr ungewöhnlich scheint, sie mit einem Sprungbefehl aufzurufen. Aber die Marken CI und CO werden mit CALL-Befehlen angesprochen und somit stimmt die Unterprogrammaufruffolge wieder.

Das Programm selbst entspricht ziemlich genau dem Flußdiagramm. Das Registerpaar HL wird als Programmzähler verwendet. Dieses wird nach dem Start mit der Adresse des Programmfeldes besetzt. Danach erfolgt der Eintritt in die Schleife LOOP. Der Befehl wird in den Akku des Prozessors 8080 geholt und dann HL um eins erhöht, so daß das Registerpaar nun auf den Parameter des zuvor eingelesenen Wertes zeigt. Dieser Parameter wird in das Register E eingelesen und HL erneut um

```
                          .phex
                          .pabs
0100                      .loc 100h
                          ;
                          ;************************
                          ;* Mini Interpreter     *
                          ;* RDK 800214           *
                          ;************************
                          ;
0100     C3 0109          Start: jmp starte
0103     C3 BE09          ci:    jmp 0be09h          ;Consol Eingabe in A
0106     C3 BE0C          co:    jmp 0be0ch          ;Consol Ausgabe in C
                          ;
                          ;
                          ;
0109                      starte:                    ;Interpreter Hauptprogramm
0109     21 1000          lxi h,prg                  ;festgelegte Prog Anfadresse
010C                      loop:                      ;Hauptschleife
010C     7E               mov a,m                    ;befehl holen
010D     23               inx h                      ;
010E     5E               mov e,m                    ;
010F     23               inx h
0110     1611             mvi d,data                 ;highorder = const data
0112     FE4C             cpi "L"                    ;Zeichen L dann LOAD
0114     CA 014C          jz load
0117     FE53             cpi "S"                    ;STORE
0119     CA 0153          jz store
011C     FE41             cpi "A"                    ;ADD
011E     CA 015A          jz add
0121     FE4E             cpi "N"                    ;NEG
0123     CA 0166          jz negate
0126     FE4A             cpi "J"                    ;JUMP
0128     CA 0170          jz jump
012B     FE49             cpi "I"                    ;JMPNZ
012D     CA 0176          jz jmpnz
0130     FE4F             cpi "O"                    ;OUTHEX
0132     CA 0180          jz outhex
0135     FE47             cpi "G"                    ;GETHEX
0137     CA 0187          jz gethex
013A     FE43             cpi "C"                    ;CRLF
013C     CA 0197          jz crlf
013F     FE5A             cpi "Z"                    ;ZEICH
0141     CA 01A4          jz zeich
0144     FE48             cpi "H"                    ;HALT
0146     CA 01AC          jz halt
0149     C3 01E4          jmp error                  ;
                          ;
                          ;einzelne Befehle
                          ;
014C                      load:
014C     1A               ldax d                     ;data laden
014D     32 021E          sta akku                   ;in virtuellen akku
0150     C3 010C          jmp loop                   ;fertig
                          ;
                          ;
0153                      store:
0153     3A 021E          lda akku                   ;holen akku
0156     12               stax d                     ;abspeichern
0157     C3 010C          jmp loop
```

Abb. 2.1-3 Listing des Mini-Interpreters

```
                        ;
                        ;
                        ;
015A                    add:
015A    1A              ldax D          ;laden wert
015B    47              mov b,a
015C    3A 021E         lda akku        ;+akku
015F    80              add b
0160    32 021E         sta akku        ;nach akku
0163    C3 010C         jmp loop
                        ;
                        ;
0166                    negate:
0166    3A 021E         lda akku
0169    2F              cma
016A    32 021E         sta akku        ;1 er Komplement
016D    C3 010C         jmp loop
                        ;
                        ;
0170                    jump:
0170    1610            mvi d,prgh      ;nur innerhalb prg
0172    EB              xchg            ;neuer pc
0173    C3 010C         jmp loop
                        ;
                        ;
0176                    jmpnz:
0176    3A 021E         lda akku
0179    B7              ora a           ;falls null nicht springen
017A    CA 010C         jz loop
017D    C3 0170         jmp jump        ;wie jump weiter
                        ;
                        ;
0180                    outhex:
0180    1A              ldax d          ;wert laden
0181    CD 01B8         call prac       ;ausgeben
0184    C3 010C         jmp loop
                        ;
                        ;
                        ;
0187                    gethex:
0187    CD 01D0         call getdig
018A    87              add a
018B    87              add a
018C    87              add a
018D    87              add a
018E    47              mov b,a
018F    CD 01D0         call getdig
0192    80              add b
0193    12              stax d
0194    C3 010C         jmp loop
                        ;
                        ;
                        ;
0197                    crlf:
0197    0E0D            mvi c,0dh
0199    CD 0106         call co
019C    0E0A            mvi c,0ah
019E    CD 0106         call co
01A1    C3 010C         jmp loop
                        ;
```

Abb. 2.1-3

```
                              ;
01A4                          zeich:
01A4        1A                ldax d              ;Zeichen ausgabe
01A5        4F                mov c,a
01A6        CD 0106           call co
01A9        C3 010C           jmp loop
                              ;
                              ;
01AC                          halt:
01AC        11 020F           lxi d,hmess         ;meldung an console
01AF        CD 01F0           call print
01B2        CD 0103           call ci
01B5        C3 0100           jmp start           ;zeichen eingeben dann rset
                              ;
                              ;
                              ; weitere Unterprogramme
                              ;
01B8                          prac:
01B8        F5                push psw
01B9        1F                rar
01BA        1F                rar
01BB        1F                rar
01BC        1F                rar
01BD        CD 01C1           call outh
01C0        F1                pop psw
01C1                          outh:
01C1        E60F              ani 0fh
01C3        C630              adi "0"
01C5        FE3A              cpi "9"+1
01C7        DA 01CC           jc outch
01CA        C627              adi "a"-"9"-1
01CC                          outch:
01CC        4F                mov c,a
01CD        C3 0106           jmp co
                              ;
                              ;
01D0                          getdig:
01D0        CD 0103           call ci
01D3        4F                mov c,a
01D4        CD 0106           call co
01D7        FE30              cpi "0"
01D9        38F5              jrc getdig
01DB                          conv:
01DB        FE3A              cpi "9"+1
01DD        3802              jrc nosub
01DF        D607              sui 7
01E1                          nosub:
01E1        E60F              ani 0fh
01E3        C9                ret
                              ;
                              ;
01E4                          error:
01E4        11 01FB           lxi d,errmess
01E7        CD 01F0           call print
01EA        CD 0103           call ci             ;nach zeichen weiter rset
01ED        C3 0100           jmp start
                              ;
                              ;
01F0                          print:
01F0        1A                ldax d
01F1        13                inx d
```

Abb. 2.1-3

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                PAGE 4
.MAIN. -


    01F2    B7              ora a
    01F3    C8              rz
    01F4    4F              mov c,a
    01F5    CD 0106         call co
    01F8    C3 01F0         jmp print
                            ;
    01FB                    errmess:
    01FB    554E44454649    .asciz "UNDEFIN\
    0201    4E4544204F50    \ED OP \
    0207    20434F4445      \CODE
    020C    0D0A00          "
    020F                    hmess:
    020F    48414C542052    .asciz "HALT RE\
    0215    455345543D59    \SET=Y
    021B    0D0A00          "
                            ;
    021E    00              akku: .byte 0     ;ram zelle
                            ;
                            ;
                            ;
                            ;daten definitionen


    1000                    prg=1000h         ;programm auf 1000 bis 10ff max
    0010                    prgh=10h

    0011                    data=11h          ;daten von 1100 bis 11ff max
                            ;
                            ;
    1000                    .loc 1000h        ;testprogramm
                            ;
    1000    4700            .byte "G",0       ;zwei Bytes  einlesen
    1002    5A01            .byte "Z",1       ;Ausgabe + Zeichen
    1004    4C00            .byte "L",0       ;Lade Wert
    1006    4700            .byte "G",0       ;Weiteren Wert Holen
    1008    4100            .byte "A",0       ;Addieren
    100A    5A02            .byte "Z",2       ;Zeichen =
    100C    5300            .byte "S",0       ;Ergebnis
    100E    4F00            .byte "O",0       ;Ausgabe
    1010    4300            .byte "C",0       ;CRLF
    1012    4A00            .byte "J",0       ;Sprung nach 0
                            ;
    1100                    .loc 1100H
    1100    00              .byte 0
    1101    2B              .byte "+"
    1102    3D              .byte "="
                            ;
                            .end
```

Abb. 2.1-3

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 5
.MAIN. -
+++++ SYMBOL TABLE +++++

ADD     015A         AKKU    021E          CI      0103
CO      0106         CONV    010B          CRLF    0197
DATA    0011         ERRMES  01FB          ERROR   01E4
GETDIG  01D0         GETHEX  0187          HALT    01AC
HMESS   020F         JMPNZ   0176          JUMP    0170
LOAD    014C         LOOP    010C          NEGATE  0166
NOSUB   01E1         OUTCH   01CC          OUTH    01C1
OUTHEX  0180         PRAC    0188          PRG     1000
PRGH    0010         PRINT   01F0          START   0100
STARTE  0109         STORE   0153          ZEICH   01A4
.BLNK.  0000:03  X   .DATA.  0000*    X    .PROG.  0000'   X
```

Abb. 2.1-3

eins erhöht, so daß HL nun auf den nächsten Befehl im Programmspeichergebiet zeigt. Das Register D wird mit der höherwertigen Adresse des Datenbereiches geladen, da mit dem Parameter zusammen normalerweise ein vom Programm getrennter Bereich adressiert wird. Nun wird der Vergleich mit dem Befehlsvorrat durchgeführt. Stimmt der im Register A vorhandene Befehl mit einem der abgefragten überein, so wird ein Sprung auf die dazugehörige Routine durchgeführt.

LOAD: Mit dem Befehl LDAX D wird der durch DE adressierte Wert in den Akku geladen. Er wird dann mit STA AKKU in den Akkumulator der hypothetischen Maschine gespeichert; anschließend findet ein Sprung zur Marke LOOP statt, und der Befehlszyklus wiederholt sich.

STORE: Der Inhalt der Speicherzelle AKKU wird in der Speicherzelle die durch DE adressiert ist abgelegt.

ADD: Es wird der Inhalt der durch DE adressierten Speicherzelle in das Register A geladen und anschließend nach Register B gerettet. Dann wird der Inhalt der Zelle AKKU in das Register A (Akku des 8080) geladen und mit dem Befehl ADD B die Addition ausgeführt. Das Ergebnis wird dann wieder in die Zelle AKKU zurückgespeichert. Hier ist ganz typisch das Verhalten des Interpreters zu sehen, daß sofort nach Erkennen des Befehls dieser ausgeführt wird und danach der nächste Befehl interpretiert wird.

JUMP: Hier wird der Inhalt des Registers D neu besetzt. Es wird der höherwertige Teil des Programmspeichergebiets in ihm abgelegt. Dann wird der Inhalt von DE mit HL vertauscht und dadurch der Sprung ausgeführt.

JUMPNZ: Es wird zunächst der Inhalt der Speicherzelle AKKU auf Null überprüft. Ist der Inhalt Null, so wird zur Marke LOOP gesprungen und damit der Befehlsholzyklus normal weitergeführt. Im anderen Fall wird der Sprungbefehl ausgeführt.

OUTHEX: Es wird der Inhalt der Speicherzelle, die durch DE adressiert ist, geladen und dann mit einem Unterprogramm in sedezimaler Schreibweise ausgegeben.

GETHEX: Eine sedezimale Zahl wird von der Konsole eingelesen und auf die Zelle (DE) gespeichert. Die Zahl muß dabei fest mit zwei Stellen eingegeben werden. Die Konvertierung von ASCII nach binär geschieht dabei für jede Stelle mit der Routine GETDIG.

Alle weiteren Routinen sind ähnlich aufgebaut. Am Schluß des Programms befindet sich noch ein kleines Programmbeispiel in der hypothetischen Sprache

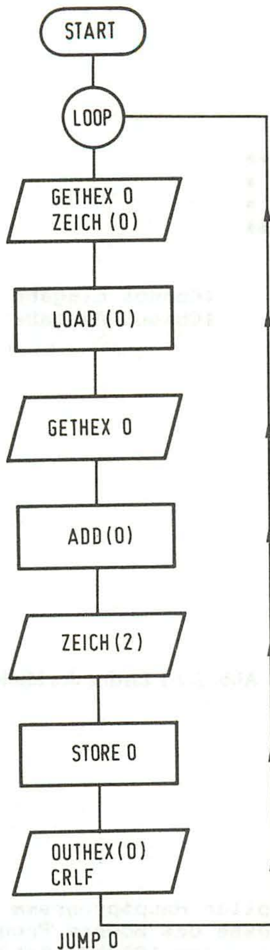


Abb. 2.1-4 **Programmbeispiel der hypothetischen Sprache**

(Abb. 2.1-4). Das Programm steht dabei auf der Adresse 1000H und die Daten auf dem Bereich 1100H. Das Programm hat die Aufgabe, zwei Bytes sedezimal einzulesen und die Summe auf der Konsole auszugeben. Bei dem Schreiben der Programme ist darauf zu achten, daß ein Befehl aus zwei Bytes besteht, dem Befehlscode und dem Parameter und damit zwei Speicheradressen belegt werden. Bei Sprüngen ist stets ein gerader Adreßwert anzugeben.

2.2 Der Compiler

Die Arbeitsweise eines Compilers soll hier anhand der gleichen hypothetischen Sprache gezeigt werden, wie sie schon im Abschnitt 2.1 für den Interpreter verwendet wurde. Der grundlegende Unterschied des Compilers zum Interpreter ist, daß die höhere Sprache nicht Schritt für Schritt ausgeführt wird, sondern zunächst einmal komplett in Maschinensprache übersetzt wird, und erst beim Start des so erzeugten neuen Programms wird das in der höheren Sprache geschriebene Programm ausgeführt.

Der Vorteil eines Compilers liegt in der schnelleren Ausführungszeit des Programms der höheren Programmiersprache. Beim Interpreter muß jedesmal, wenn ein neuer Befehl der Sprache erscheint, dieser zunächst erkannt und dann das dazugehörige Unterprogramm aufgerufen werden. Beim Compiler wird dieser Analyseprozeß nur einmal bei der Übersetzung durchgeführt und beim Starten des erzeugten Maschinenprogramms ist eine Analyse nicht mehr nötig. Dieser Unterschied fällt besonders bei Schleifen auf: Der Interpreter muß mit jedem Durchlauf das Programm analysieren, der Compiler analysiert die gesamte Schleife nur einmal – unabhängig davon, wie oft die Schleife eigentlich durchlaufen werden müßte. Bei unserer hypothetischen Sprache fällt dieser Unterschied zwar nicht so sehr ins Gewicht, aber immerhin sind die mit dem Compiler erzeugten Programme deutlich schneller als die durch den Interpreter ausgeführten. *Abb. 2.2-1* zeigt das Listing des Compilers. Der Compiler beginnt zunächst mit der Sprungtabelle für die I/O-Vektoren. Sie sind je nach vorhandenem Computer entsprechend anzupassen. Danach erfolgt die Definition eines Makros zur Vereinfachung des Assemblerprogramms. Wer noch nie mit Makros gearbeitet hat, dem sei hier eine kurze Erklärung gegeben:

Ein Makro kann mit einem Unterprogrammaufruf verglichen werden. Der

```
                              .phex
                              .pabs
0100                          .loc 100h

                              ;*********************
                              ;* Mini Compiler     *
                              ;* RDK 800316        *
                              ;*********************

0100        C3 0115           Start: jmp starte
0103        C3 BE09           ci:    jmp 0be09h          ;Consol Eingabe in A
0106        C3 BE0C           co:    jmp 0be0ch          ;Consol Ausgabe in C
                              ; Macro Definitionen

                              .define PRINT[a,%b]=
                              [
                              lxi b,.+8
                              call lprint
                              jmpr %b
                              .ascizs a
                              %b:
                              ]
0109                          lprint:
0109        0A                ldax b
010A        B7                ora a
010B        C8                rz
010C        C5                push b
010D        4F                mov c,a
010E        CD 0106           call co
0111        C1                pop b
0112        03                inx b
0113        18F4              jmpr lprint


0115                          starte:                    ;Compiler Hauptprogramm
0115        21 1000           lxi h,prg                  ;Adresse des Source Programms
0118        01 0120           print '.insert   runtime.asm '[0dh] [0ah]
011B        CD 0109
011E        1818
0120        2E696E736572
0126        74202072756E
012C        74696D652E61
0132        736D200D0A00
0138                          loop:
0138        EB                xchg
0139        01 0141           print "L"
013C        CD 0109
013F        1802
0141        4C00
0143        AF                xra a
0144        CD 03D7           call prac
0147        7B                mov a,e
0148        CD 03D7           call prac
014B        01 0153           print "H:"[0dh] [0ah]
014E        CD 0109
0151        1805
0153        483A0D0A00
0158        EB                xchg
```

Abb. 2.2-1 Listing des Mini-Compilers

```
0159    7E              mov a,m         ;Befehl holen

015A    23              inx h
015B    5E              mov e,m             ;Parameter
015C    23              inx h
015D    1611            mvi d,data      ;fuer constanten
015F    FE4C            cpi "L"
0161    CA 019E         jz load
0164    FE53            cpi "S"
0166    CA 01C4         jz store
0169    FE41            cpi "A"
016B    CA 01EA         jz add
016E    FE4E            cpi "N"
0170    CA 0245         jz negate
0173    FE4A            cpi "J"
0175    CA 027C         jz jump
0178    FE49            cpi "I"
017A    CA 0292         jz jmpnz
017D    FE4F            cpi "O"
017F    CA 02CB         jz outhex
0182    FE47            cpi "G"
0184    CA 0309         jz gethex
0187    FE43            cpi "C"
0189    CA 02F2         jz crlf
018C    FE5A            cpi "Z"
018E    CA 0332         jz zeich
0191    FE48            cpi "H"
0193    CA 0369         jz halt
0196    FE45            cpi "E"         ;Ende zeichen
0198    CA 03A5         jz final
019B    C3 0404         jmp error           ;falsches Zeichen


                        ; einzelne Befehle

019E                    load:
019E    01 01A6         print "lda "
01A1    CD 0109
01A4    1805
01A6    6C64612000
01AB    CD 03EF         call outde          ;parameter
01AE    01 01B6         print "sta akku" [0dh] [0ah]
01B1    CD 0109
01B4    180B
01B6    73746120616B
01BC    6B750D0A00
01C1    C3 0138         jmp loop

01C4                    store:
01C4    01 01CC         print "lda akku" [0dh] [0ah]
01C7    CD 0109
01CA    180B
01CC    6C646120616B
01D2    6B750D0A00
01D7    01 01DF         print "sta "
01DA    CD 0109
01DD    1805
01DF    7374612000
01E4    CD 03EF         call outde
01E7    C3 0138         jmp loop
```

Abb. 2.2-1

Abb. 2.2-1

```
01EA                    add:
01EA    01 01F2         print "lda "
01ED    CD 0109
01F0    1805
01F2    6C64612000
01F7    CD 03EF         call outde
01FA    01 0202         print 'mov b a'[0dh] [0ah]
01FD    CD 0109
0200    180A
0202    6D6F7620622
0208    610D0A00
020C    01 0214         print "lda akku"[0dh] [0ah]
020F    CD 0109
0212    180B
0214    6C646120616B
021A    6B750D0A00
021F    01 0227         print "add b"[0dh] [0ah]
0222    CD 0109
0225    1808
0227    61646420620D
022D    0A00
022F    01 0237         print "sta akku"[0dh] [0ah]
0232    CD 0109
0235    180B
0237    73746120616B
023D    6B750D0A00
0242    C3 0138         jmp loop

0245                    negate:
0245    01 024D         print "lda akku"[0dh] [0ah]
0248    CD 0109
024B    180B
024D    6C646120616B
0253    6B750D0A00
0258    01 0260         print "cma"[0dh] [0ah]
025B    CD 0109
025E    1806
0260    636D610D0A00
0266    01 026E         print "sta akku"[0dh] [0ah]
0269    CD 0109
026C    180B
026E    73746120616B
0274    6B750D0A00
0279    C3 0138         jmp loop

027C                    jump:
027C    1600            mvi d,0         ;labels 00xxh
027E    01 0286         print "jmp L"
0281    CD 0109
0284    1806
0286    6A6D70204C00
028C    CD 03EF         call outde      ;labelnummer schreiben
028F    C3 0138         jmp loop

0292                    jmpnz:
0292    01 029A         print "lda akku"[0dh] [0ah]
0295    CD 0109
0298    180B
029A    6C646120616B
02A0    6B750D0A00
02A5    01 02AD         print "ora a"[0dh] [0ah]
```

Abb. 2.2-1

```
02A8    CD 0109
02AB    1808
02AD    6F726120610D
02B3    0A00
02B5    1600            mvi d,0
02B7    01 02BF         print "jnz L"
02BA    CD 0109
02BD    1806
02BF    6A6E7A204C00
02C5    CD 03EF         call outde
02C8    C3 0138         jmp loop

02CB                    outhex:
02CB    01 02D3         print "lda "
02CE    CD 0109
02D1    1805
02D3    6C64612000
02D8    CD 03EF         call outde
02DB    01 02E3         print "call prac"[0dh] [0ah]
02DE    CD 0109
02E1    180C
02E3    63616C6C2070
02E9    7261630D0A00
02EF    C3 0138         jmp loop

02F2                    crlf:
02F2    01 02FA         print "call crlf"[0dh] [0ah]
02F5    CD 0109
02F8    180C
02FA    63616C6C2063
0300    726C660D0A00
0306    C3 0138         jmp loop

0309                    gethex:
0309    01 0311         print "call gethex"[0ah] [0dh]
030C    CD 0109
030F    180E
0311    63616C6C2067
0317    65746865780A
031D    0D00
031F    01 0327         print "sta "
0322    CD 0109
0325    1805
0327    7374612000
032C    CD 03EF         call outde
032F    C3 0138         jmp loop

0332                    zeich:
0332    01 033A         print "lda "
0335    CD 0109
0338    1805
033A    6C64612000
033F    CD 03EF         call outde
0342    01 034A         print 'mov c a'[0dh] [0ah]
0345    CD 0109
0348    180A
034A    6D6F76206320
0350    610D0A00
0354    01 035C         print "call co"[0dh] [0ah]
0357    CD 0109
035A    180A
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                                PAGE 5
.MAIN. -

   035C     63616C6C2063
   0362     6F000A00
   0366     C3 0138         jmp loop

   0369                     halt:
   0369     01 0371         print 'lxi d hmess'[0dh] [0ah]
   036C     CD 0109
   036F     180E
   0371     6C7869206420
   0377     686D6573730D
   037D     0A00
   037F     01 0387         print "call print"[0dh] [0ah]
   0382     CD 0109
   0385     180D
   0387     63616C6C2070
   038D     72696E740D0A
   0393     00
   0394     01 039C         print "hlt"[0dh] [0ah]
   0397     CD 0109
   039A     1806
   039C     686C740D0A00
   03A2     C3 0138         jmp loop

   03A5                     final:
   03A5     01 03AD         print ".end"[0dh] [0ah]
   03A8     CD 0109
   03AB     1807
   03AD     2E656E640D0A
   03B3     00

   03B4     01 03BC         print "Ende der Compilierung"[0dh] [0ah]
   03B7     CD 0109
   03BA     1818
   03BC     456E64652064
   03C2     657220436F6D
   03C8     70696C696572
   03CE     756E670D0A00


   03D4     CD F01E         call 0f01eh         ;Monitor aufrufen

   03D7                     prac:
   03D7     F5              push psw
   03D8     1F              rar
   03D9     1F              rar
   03DA     1F              rar
   03DB     1F              rar
   03DC     CD 03E0         call outh
   03DF     F1              pop psw
   03E0                     outh:
   03E0     E60F            ani 0fh
   03E2     C630            adi "0"
   03E4     FE3A            cpi "9"+1
   03E6     DA 03EB         jc outch
   03E9     C627            adi "a"-"9"-1
   03EB                     outch:
   03EB     4F              mov c,a
   03EC     C3 0106         jmp co
```

Abb. 2.2-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                        PAGE 6
.MAIN. -


     03EF                          outde:
     03EF    7A                    mov a,d
     03F0    CD 03D7               call prac
     03F3    7B                    mov a,e
     03F4    CD 03D7               call prac
     03F7    01 03FF               print "H"[0dh] [0ah]
     03FA    CD 0109
     03FD    1804
     03FF    480D0A00
     0403    C9                    ret


     0404                          error:
     0404    01 040C               print " ERROR UNDEFINED COMMAND "[0dh] [0ah]
     0407    CD 0109
     040A    181C
     040C    204552524F52
     0412    20554E444546
     0418    494E45442043
     041E    4F4D4D414E44
     0424    200D0A00
     0428    C3 0138               jmp loop

     0011                          data=11h

     1000                          .loc 1000h           ;Beispiel Programm

     1000                          prg:
     1000    4700                  .byte "G",0
     1002    5A01                  .byte "Z",1
     1004    4C00                  .byte "L",0
     1006    4700                  .byte "G",0
     1008    4100                  .byte "A",0
     100A    5A02                  .byte "Z",2
     100C    5300                  .byte "S",0
     100E    4F00                  .byte "O",0
     1010    4300                  .byte "C",0
     1012    4A00                  .byte "J",0


     1014    4701                  .byte "G",1          ;Weiteres Programm
     1016    4F01                  .byte "O",1
     1018    4C01                  .byte "L",1
     101A    4914                  .byte "I",14h
     101C    4A1C                  .byte "J",1ch

     101E    4500                  .byte "E",0

     1100                          .loc 1100h
     1100    00                    .byte 0
     1101    2B                    .byte "+"
     1102    3D                    .byte "="

                                   .end
```

Abb. 2.2-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21        Abb. 2.2.-1              PAGE 7
.MAIN. -
+++++ SYMBOL TABLE +++++

ADD     01EA          CI      0103          CO      0106          CRLF    02F2
DATA    0011          ERROR   0404          FINAL   03A5          GETHEX  0309
HALT    0369          JMPNZ   0292          JUMP    027C          LOAD    019E
LOOP    0138          LPRINT  0109          NEGATE  0245          OUTCH   03EB
OUTDE   03EF          OUTH    03E0          OUTHEX  02CB          PRAC    03D7
PRG     1000          START   0100          STARTE  0115          STORE   01C4
ZEICH   0332          .BLNK.  0000:03 X     .DATA.  0000*    X    .PROG.  0000'    X
```

Makro wird mit einem Namen versehen. Unter diesem Namen ist er im Assemblerprogramm von nun an aufrufbar; nur wird im Gegensatz zum Unterprogrammaufruf kein CALL ausgeführt, sondern der Inhalt wird in das Assemblerprogramm eingefügt. Dies ist im Prinzip genauso, als ob anstelle des Aufrufs die Befehle an die betreffende Stelle getippt würden. Im Gegensatz zum Unterprogrammauf ist es nun beim Makro möglich, in Abhängigkeit von Parametern, die dem Makroauf beigefügt werden, den eingefügten Code zu variieren. Darin liegt die Stärke bei der Verwendung von Makros. Hier wird nun ein Makro definiert, der die Aufgabe hat, wie der PRINT-Befehl in Basic zu wirken, mit der Einschränkung, daß nur Texte ausgegeben werden können.

Mit „.define" startet die Definition des Makros. Es folgt der Name mit in eckigen Klammern stehenden Parametern. Der Name lautet PRINT und die Parameter sind hier a und %b. %b ist eigentlich kein echter Parameter und dient nur der automatischen Erzeugung einer Marke. Wird innerhalb eines Makros eine Marke angegeben, so würde der Assembler beim zweiten Aufruf dieses Makros einen Fehler ausgeben, da die Marke dann zweimal deklariert ist. Es gibt auch Assembler, bei denen lokale Marken bezüglich des Makros angegeben werden können, dann ist die Marke außerhalb des vom Makros erzeugten Codes unwirksam und es tritt kein Fehler auf. Der TDL-Assembler besitzt zwar auch die Möglichkeit, lokale Marken zu definieren, doch diese sind nur lokal bezüglich zweier globaler Marken; bei dem Nacheinander zweier Makros ohne eine dazwischen liegende globale Marke tritt wieder ein Fehler auf. Daher wird hier die Möglichkeit genutzt, daß dieser Assembler automatisch Marken erzeugen kann. Durch das Zeichen „%" wird dem Assembler mitgeteilt, daß der nachfolgende Name durch eine automatisch erzeugte Marke ersetzt werden soll. Mit „=[" wird der Start des Makrokörpers angezeigt. Dort ist die Definition des eigentlichen Makros. Sie beginnt mit „lxi b,.+8;" das Registerpaar BC wird mit der Adresse des auszugebenden Textes geladen. Sie errechnet sich aus der Summe von aktuellem Programmzählerstand (mit „ . " anzugeben) und der Zahl 8, die Anzahl der dazwischen liegenden Bytes. Es erfolgt dann der Aufruf des Unterprogramms lprint, das einen Text ausgibt. Und dann folgt ein Sprung auf die Marke „%b"; sie wird dabei bei jedem neuen Makro-Aufruf variiert. Mit „.asciz a" wird der als Parameter a angegebene Text als Code abgelegt. Das Ende des Textes wird durch ein Byte mit dem Wert 0 angezeigt. Danach steht „%b:" , dies ist die Stelle der automatisch zu erzeugenden Marke. Die Definition des Makros schließt mit der eckigen Klammer. Es folgt das Unterprogramm lprint. Es hat die Aufgabe,

einen Text, dessen Anfangsadresse in DE steht, auf die Konsole auszugeben. Das Ende des Textes wird durch ein Nullbyte erkannt.

Das Hauptprogramm beginnt mit der Marke „starte". Es wird auch hier wie beim Interpreter ein Programmzähler geladen. Dieser hat jedoch hier eine andere Aufgabe. Er muß dafür sorgen, daß Marken und Sprünge, die bei unserer Programmiersprache nicht symbolisch, sondern mit einem sedezimalen Wert angegeben sind, richtig in ein Maschinenprogramm umgesetzt werden können. Da es bei uns nicht nötig ist, Marken zu definieren, werden sie hier der Einfachheit halber nach jedem Übersetzungsvorgang eines Befehls automatisch erzeugt.

Doch zunächst erfolgt der Aufruf des Makros PRINT: Es wird die erste Programmzeile ausgegeben. Soll das Programm wirklich übersetzt werden, so wäre es nötig, diesen Text nicht einfach auf eine Konsole zu geben, sondern auf eine Diskette oder sonstigen Speicher. Es wird hier nicht ein Maschinenprogramm im Objektformat, sondern als Assembler-Quellencode erzeugt. Dieses erzeugte Sourceprogramm muß nach der Compilierung von einem Assembler in die Maschinensprache übersetzt werden. Diese Vorgehensweise hat den Vorteil, daß der Compiler nicht auch noch zusätzlich einen Assembler beinhalten muß, der allerdings relativ einfach aufgebaut sein könnte.

Mit dem ersten Befehl, der erzeugt wird „ .insert runtime.asm", wird das Einfügen eines Standardprogrammpakets veranlaßt, das bei jedem erzeugten Programm benötigt wird. Dieses Paket kann zum Beispiel die Umwandlung von ASCII in Sedezimal etc. beinhalten. Aus diesem Paket werden später Unterprogramme verwendet.

Nun folgt die Hauptschleife des Compilers. Sie beginnt mit der Ausgabe einer Marke „LxxxxH:". „xxxx" wird je nach Stand des Zählers HL sedezimal ausgegeben. Diese Vorgehensweise ist hier nötig, um unnötigen Aufwand im Compiler zu umgehen. Soll nur immer dann eine Marke erzeugt werden, wenn eine benötigt wird, so muß in einem ersten Durchlauf zunächst nach Sprüngen gesucht werden. Alle darin vorkommenden Adressen werden gesammelt, und in einem zweiten Durchlauf wird nur dann eine Marke erzeugt, wenn auf die entsprechende Adresse auch tatsächlich gesprungen wird.

Im nachfolgenden Programmteil werden nun die einzelnen Befehle abgefragt. Dabei steht im Register A der Befehl und im Register E der Operand, genau wie beim Interpreter. Auch hier erfolgt ein Sprung zu der entsprechenden Routine, die den Befehl behandelt. Doch wird hier nicht das Programm ausgeführt, sondern nur der Assembler-Quellencode erzeugt. Bei LOAD z.B. wird die Sequenz „lda xxxxh ; sta akku" erzeugt. Dabei wird für „xxxx" der Parameter eingesetzt plus der Anfangsadresse des Datenbereichs. Dies geschieht in der Routine „outde".

Auf diese Art und Weise wird für jeden Befehl eine Sequenz von Maschinenbefehlen erzeugt. Zu beachten ist, daß auch beim bedingten Sprung nicht etwa der Akkumulator abgefragt wird, sondern stur die Befehlsfolge erzeugt wird, unabhängig vom Zustand eines Registers der hypothetischen Sprache.

Die Abfrage des Akkumulators erfolgt erst beim Starten des erzeugten Maschinenprogramms. Auch das Verhalten bei dem HALT-Befehl ist anders. Es wird keine Meldung an die Konsole ausgegeben, sondern hier tatsächlich ein HALT-Befehl des 8080 in das erzeugte Programm geschrieben. Dies führt später zum Stopp des Prozessors, wenn er an diese Stelle kommt, doch der Übersetzungsvorgang wird nicht gestoppt. Er kann nur durch die Angabe eines sogenannten Pseudobefehls gestoppt werden. Dies ist ein Befehl, der in der Maschinen-Sprache eigentlich keine Bedeutung hat. Er dient nur dazu, dem Compiler das Ende des Programms mitzuteilen. Beim Interpre-

```
BASCOM 5.1 - Copyright 1979 (C) by MICROSOFT - 14769 Bytes Free
 0014 0007     10 REM BEISPIEL FUER DEN BASIC
         ** 0014'L00010:
 0014 0007      20 REM COMPILER
         ** 0014'L00020:
 0014 0007     30 FOR I=1 TO 10
         ** 0014'L00030: CALL     $LFMA
         ** 0017'        DW       <const>
         ** 0019'        JMP      I00000
         ** 001C'I00001:
 001C 0007      40 PRINT I,I*I
         ** 001C'L00040: CALL     $PR0A
         ** 001F'        LXI      H,I!
         ** 0022'        CALL     $PV0A
         ** 0025'        CALL     $FMUA
         ** 0028'        DW       I!
         ** 002A'        DW       I!
         ** 002C'        LXI      H,$AC%
         ** 002F'        CALL     $PV2A
 0032 000B      50 NEXT I
         ** 0032'L00050: CALL     $FADA
         ** 0035'        DW       I!
         ** 0037'        DW       <const>
         ** 0039'I00000:
         ** 0039'        CALL     $FASO
         ** 003C'        DW       I!
         ** 003E'        CALL     $LEJA
         ** 0041'        DW       I!
         ** 0043'        DW       <const>
         ** 0045'        DW       I00001
 0047 000B       60 END
         ** 0047'L00060: CALL     $END
 004A 000B
         ** 004A'        CALL     $END

00000 Fatal Error(s)
14532 Bytes Free


 L80 B:BASICC1 /G


[0118    234E    35]
[Begin execution]
 1         1
 2         4
 3         9
 4         16
 5         25
 6         36
 7         49
 8         64
 9         81
 10        100
```

Abb. 2.2-3 Basic-Compiler-Ausdruck

Abb. 2.2-2 Ausgabebeispiel des Compilers

```
.insert  runtime.asm
L0000H:
call gethex
sta 1100H
L0002H:
lda 1101H
mov c a
call co
L0004H:
lda 1100H
sta akku
L0006H:
call gethex
sta 1100H
L0008H:
lda 1100H
mov b a
lda akku
add b
sta akku
L000aH:
lda 1102H
mov c a
call co
L000cH:
lda akku
sta 1100H
L000eH:
lda 1100H
call prac
L0010H:
call crlf
L0012H:
 jmp L0000H
L0014H:
call gethex
sta 1101H
L0016H:
lda 1101H
call prac
L0018H:
lda 1101H
sta akku
L001aH:
lda akku
ora a
jnz L0014H
L001cH:
jmp L001cH
L001eH:
.end
Ende der Compilierung
```

ter war dieser Befehl nicht nötig. *Abb. 2.2-2* zeigt noch ein Ausgabebeispiel des Compilers.

Abb. 2.2-3 zeigt das Beispiel eines Basic-Compilers. Der Aufruf L80 ... nach der Übersetzung des Programms bewirkt den Start eines Linkers, der aus einer Bibliothek Standard-Unterprogramme, wie Gleitkommaarithmetik dazu bindet. Danach erst kann das Programm gestartet werden. Der Aufruf dieser Unterprogramme ist im Übersetzungsprotokoll durch Namen die mit „$" beginnen, gekennzeichnet.

Abb. 2.3-1 Übersetzungstabelle

2.3 Teilübersetzung als Kompromiß

Bei den meisten handelsüblichen Basic-Interpretern wird ein Kompromiß zwischen der Version aus Kapitel 2.1 und einem Compiler gezogen. Manche dieser Interpreter nennen sich sogar Compiler; das allerdings zu Unrecht.

Bei der Analyse und Erkennung auf ein Wort wird relativ viel Zeit verbraucht. Daher wird nun die Wortanalyse gleich beim Ablegen des Programms vorgenommen und anstatt des Basic-Wortes PRINT z.B. der Hexadezimal-Code 81 im Speicher abgelegt. Dadurch wird nicht nur Platz im Speicher gespart, sondern es ist bei der nachfolgenden Interpretation des Basic-Programms viel leichter möglich, diese Wörter zu erkennen. In diesem Beispiel muß nur das höchstwertige Bit 7 abgefragt werden, und schon ist dem Interpreter bekannt, ob es sich um ein Basic-Wort handelt oder nicht. Mit den restlichen 7 Bits kann über eine Adreßtabelle mit einem indirekten Sprung das entsprechende Unterprogramm ausgeführt werden. *Abb. 2.3-1* zeigt eine Übersetzungstabelle, wie sie in einem Basic-Interpreter vorhanden sein könnte.

Die Basic-Wörter werden immer gefolgt von dem Code, durch den sie bei der Übersetzung ersetzt werden. Der Scanner, so wird dieser Vorübersetzer genannt, sucht nun in dieser Tabelle nach dem Wort und falls er es findet, wird der nachstehende Code im Speicher abgelegt. Der Code muß natürlich so gewählt werden, daß später Verwechslungen ausgeschlossen sind. Bei ASCII (7 Bits, 0...6) können für diesen Code einfach Zeichen mit gesetztem Bit 7 verwendet werden. Bei manchen Basic-Interpretern, so zum Beispiel bei PET und CBM, müssen in Strings auch Graphik-Zeichen vorkommen können; dies sind dort ebenfalls Zeichen mit gesetztem Bit 7. Eine Verwechslung ist dennoch ausgeschlossen, da Strings immer mit Anführungszeichen anfangen müssen und auch mit diesem Zeichen (") enden. Dadurch schaltet der Interpreter um und betrachtet alle Zeichen zwischen zwei Anführungszeichen als Text, ohne auf die Spezialcodes zu sehen.

Abb. 2.3-2 veranschaulicht den Kompressionsvorgang. Das reservierte Wort PRINT wird im Speicher durch den Code 81H dargestellt. Der Rest der Zeile läßt sich nicht vereinfachen. Unnötige Leerzeichen können hier nicht entfernt werden, da später beim LIST-Befehl diese Zeichen wieder erscheinen müssen.

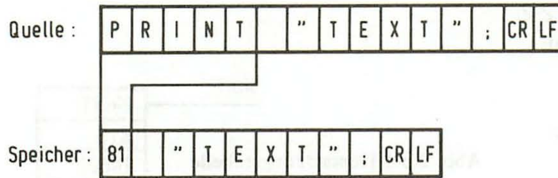


Abb. 2.3-2 Kompressionsvorgang

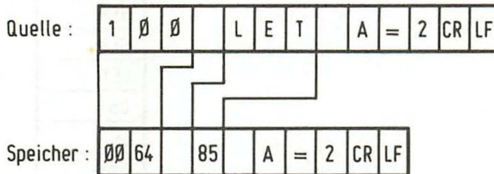


Abb. 2.3-3 Kompressionsvorgang mit Zeilennummer

Ein ähnlicher Kompressionsvorgang läßt sich mit der Zeilennummer durchführen. *Abb. 2.3-3* zeigt das Beispiel. Die Zeilennummer wird binär codiert und so direkt im Speicher abgelegt. Sie belegt dadurch 2 Bytes (daher auch die Beschränkung der meisten Basic-Interpreter auf ca. 32000 bzw. 64000 als höchste Zeilennummer). Die Ausführungsgeschwindigkeit wird dadurch stark erhöht, da der Dezimal-Binär-Umwandlungsvorgang nicht jedesmal beim Ausführen einer Zeile, sondern nur einmal durchgeführt werden muß.

Ebenfalls wäre die Ausführung eines GOTO-Befehls sonst eine recht zeitraubende Angelegenheit. Manche Basic-Interpreter gehen noch etwas weiter und fügen hinter die binäre Zeilennummer noch die Adresse der nächsten Zeile, die sonst nur durch das Zeichen CR (Carriage Return) auffindbar wäre. Dadurch wird der Suchvorgang für das Zeilenende eingespart.

3 RDK-Basic

3.1 Ein „Tiny Basic" und dessen Einsatz

In diesem Kapitel soll der praktische Aufbau eines kompletten Basic-Interpreters gezeigt werden. Er entspricht einem „Tiny Basic", also einem Basic mit eingeschränktem Befehlsvorrat. Der Begriff „tiny" tritt gerade in der amerikanischen Fachliteratur sehr häufig auf. So gibt es mittlerweile neben einem Tiny Basic auch ein „Tiny Pascal". Der Begriff „tiny" (winzig) bedeutet, daß es sich um einen „Subset", also eine Untermenge der eigentlichen Programmiersprache handelt. Im Falle von Basic heißt dies meist, daß eine Gleitkomma-Arithmetik fehlt und die Fähigkeit, Strings (Zeichenketten) zu verarbeiten, zumindest nicht so wie es im Original-Basic definiert ist.

Auch hier soll nur der Interpreter eines Tiny Basic besprochen werden, da dieser den großen Vorteil bietet, sehr wenig Speicherplatz zu belegen und dadurch für eine Programmbesprechung einigermaßen übersichtlich bleibt. Tiny-Basic-Interpreter belegen im allgemeinen einen Speicherraum von 1,5...4 KByte, ein Standard-Basic etwa 5...8 KByte und ein erweitertes Basic mit komfortabler Stringbehandlung und speziellen Befehlen 12 KByte, Disk-Basic schließlich 16...32 KByte und Compiler sogar etwa 16...48 KByte (bei 8080 oder Z80).

Der Vorteil eines Tiny Basic liegt neben dem geringen Speicherbedarf auch darin, daß er sehr schnell arbeitet, da er eine Ganzzahl-Arithmetik verwendet, die die CPU meist von Haus aus recht gut beherrscht. Eine Gleitkomma-Arithmetik wie sie in vielen Interpretern ausschließlich verwendet wird, verlangsamt die Ausführung eines Basic-Programms. In Fällen also, wo keine Gleitkomma-Arithmetik nötig ist, ist das Arbeiten mit Tiny Basic vorteilhafter als mit Standard-Basic. Ein Einsatz ist also in den Bereichen Prozeßdatenverarbeitung, Grafik und Spiele zu suchen.

Das hier beschriebene Basic wurde ursprünglich in einer amerikanischen Zeitschrift [11] veröffentlicht; die Länge betrug 1,75 KByte. Es wurde stark vergrößert und neue Funktionen wurden aufgenommen: Die jetzige Länge beträgt 3 KByte, und das Programm liegt in dieser Form dem Leser vor.

3.2 Hardware bei Z80 und Z8000 zum Betrieb des Tiny Basic

Das Programm war ursprünglich für den 8080 ausgelegt. Der Code ist nach wie vor für den 8080 gedacht, doch kann natürlich auch ein Z80 diesen Code ausführen. Die entsprechenden Stellen, die inkompatibel sein könnten (P/V-Flag), wurden bei der Programmierung berücksichtigt.

Das gleiche Programm liegt auch in einer Version für den Z8000 vor, wobei entsprechende Optimierungen im Hinblick auf die Leistungsfähigkeit des Z8000-Befehlssatzes durchgeführt wurden. Das Listing für den Z8000 ist genauso gegliedert, so daß in der Besprechung hauptsächlich auf das 8080/Z80-Listing zurückgegriffen und nur in besonderen Fällen auf den Z8000 eingegangen wird.

Das Basic läßt sich auf fast jedem 8080/Z80-Rechner installieren, zum Betrieb sind nur drei Sprungvektoren zu ändern, die auf Konsolroutinen zeigen. Beim Z8000 ist dies im Prinzip ähnlich, nur wird dort im allgemeinen mit einem System-CALL-Befehl gearbeitet, um mit Konsolen und anderen Peripherieeinheiten zu arbeiten. Um trotzdem Änderungen leicht durchführen zu können, wurde innerhalb des Basic mit CALL-Befehlen gearbeitet, die eine Stelle am Anfang des Basic aufrufen; diese Stelle enthält dann die „System-Calls". Ferner wurde noch ein Monitorprogramm beigefügt, das den Aufbau eines Z8000-Systems erleichtern soll. Die dazugehörige Hardware ist zusammen mit dem Monitorprogramm in [6] beschrieben. Das Listing ist im Anhang abgedruckt. Weitere Hardware findet sich in [5].

3.3 Realisierung des Basic-Interpreters

In diesem Abschnitt wird anhand des Listings von *Abb. 3.3-1* der Aufbau und die Wirkungsweise eines Basic-Interpreters erklärt. Dabei unterteilt sich der Interpreter in unterschiedliche Module, die entsprechend ihres Schwierigkeitsgrades in verschiedenen Abschnitten besprochen werden. *Abb. 3.3-2* zeigt den Ausdruck in sedezimaler Form.

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 1
.MAIN. -

                              .PABS
                              .PHEX
                              ;*************************************
                              ; RDK BASIC INTERPRETER V3.2 800316
                              ;*************************************
1000                          ANBAS=\ " ANFANGS ADRESSE BASIC"

0100                          .LOC 100H           ;BEI CPM
0100      C3 0FE3             JMP BEGINN
0FE3                          .LOC ANBAS-29
0FE3                          BEGINN:
0FE3      31 210F             LXI SP,STACK
0FE6      3EC9                MVI A,0C9H
0FE8      32 1000             STA HAUPTP
0FEB      CD 1000             CALL HAUPTP
0FEE                          ANF:
0FEE      3B                  DCX SP
0FEF      3B                  DCX SP
0FF0      D1                  POP D
0FF1      21 0012             LXI H,HAUPTP-ANF
0FF4      19                  DAD D
0FF5      11 1000             LXI D,HAUPTP
0FF8      01 0A86             LXI B,ENDE-BEGINN
0FFB      ED80                LDIR
0FFD      C3 16C0             JMP START

                              ;HAUPTPROGRAMM
1000                          HAUPTP:
1000      C3 16C0             JMP START
1003      C3 10B8             JMP RSTART
```

Abb. 3.3-1 Listing des RDK-BASIC

```
1006    C3 F003     CI:JMP 0F003H
1009    C3 F009     ECHO:JMP 0F009H
100C    C3 F012     CSTS:JMP 0F012H
100F                COMP:
100F    7C          MOV A,H
1010    BA          CMP D
1011    C0          RNZ
1012    7D          MOV A,L
1013    BB          CMP E
1014    C9          RET

1015                IGNB:
1015    1A          LDAX D
1016    FE20        CPI " "
1018    C0          RNZ
1019    13          INX D
101A    C3 1015     JMP IGNB

101D                FINI:
101D    F1          POP PSW
101E    CD 150E     CALL FIN
1021    C3 1527     JMP QWHAT

1024                TSTV:
1024    CD 1015     CALL IGNB
1027    D640        SUI 040H
1029    D8          RC
102A    C2 1048     JNZ TV1
102D    13          INX D


TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                        PAGE 2
.MAIN. - RDK  B A S I C  INTERPRETER 3K  780524  V3.3


102E    CD 1462     CALL PARN
1031    29          DAD H
1032    DA 1096     JC QHOW
1035    D5          PUSH D
1036    EB          XCHG
1037    CD 14A9     CALL SIZE
103A    CD 100F     CALL COMP
103D    DA 1557     JC ASORRY
1040    2A 2148     LHLD TXTEND
1043    CD 14CC     CALL SUBDE
1046    D1          POP D
1047    C9          RET
1048                TV1:
1048    FE1B        CPI 01BH
104A    3F          CMC
104B    D8          RC
104C    13          INX D
104D    21 2111     LXI H,VARBGN
1050    07          RLC
1051    85          ADD L
1052    6F          MOV L,A
1053    3E00        MVI A,0
1055    8C          ADC H
1056    67          MOV H,A
1057    C9          RET
```

Abb. 3.3-1

```
1058                   TSTC:
1058    E3             XTHL
1059    CD 1015        CALL IGNB
105C    BE             CMP M
105D                   TC1:
105D    23             INX H
105E    CA 1068        JZ TC2
1061    C5             PUSH B
1062    4E             MOV C,M
1063    0600           MVI B,0
1065    09             DAD B
1066    C1             POP B
1067    1B             DCX D
1068                   TC2:
1068    13             INX D
1069    23             INX H
106A    E3             XTHL
106B    C9             RET

106C                   TSTNUM:
106C    21 0000        LXI H,0
106F    44             MOV B,H
1070    CD 1015        CALL IGNB
1073                   TN1:
1073    FE30           CPI "0"
1075    D8             RC
1076    FE3A           CPI 03AH
1078    D0             RNC
1079    3EF0           MVI A,0F0H
107B    A4             ANA H
107C    C2 1096        JNZ QHOW
107F    04             INR B
1080    C5             PUSH B
1081    44             MOV B,H
1082    4D             MOV C,L


TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 3
.MAIN. - RDK  B A S I C  INTERPRETER 3K  780524  V3.3


1083    29             DAD H
1084    29             DAD H
1085    09             DAD B
1086    29             DAD H
1087    1A             LDAX D
1088    13             INX D
1089    E60F           ANI 0FH
108B    85             ADD L
108C    6F             MOV L,A
108D    3E00           MVI A,0
108F    8C             ADC H
1090    67             MOV H,A
1091    C1             POP B
1092    1A             LDAX D
1093    F2 1073        JP TN1
1096                   QHOW:
1096    D5             PUSH D
1097                   AHOW:
1097    11 109D        LXI D,HOW
109A    C3 152B        JMP ERROR
```

Abb. 3.3-1

```
109D                    HOW:
109D    484F573F        .ASCII /HOW?
10A1    0D0A            /
10A3                    OK:
10A3    5245414459      .ASCII /READY
10A8    0D0A            /
10AA                    WHAT:
10AA    574841543F      .ASCII /WHAT?
10AF    0D0A            /
10B1                    SORRY:
10B1    534F525259      .ASCII /SORRY
10B6    0D0A            /
                        ;*************************************
                        ;    HAUPTPROGRAMM
                        ; LEGT PROGRAMM IM SPEICHER AB
                        ; DEFINIERT REGISTER
                        ;LIEST EINE BENUTZERZEILE UM
                        ;DIESE ZU VERARBEITEN
                        ;*************************************
                        ;
10B8                    RSTART:
10B8    31 210F         LXI SP,STACK
10BB                    ST1:
10BB    CD 16F9         CALL CRLF
10BE    11 10A3         LXI D,OK
10C1    97              SUB A
10C2    CD 15CA         CALL PRTSTG
10C5    21 10CC         LXI H,ST2+1
10C8    22 2007         SHLD CURRNT
10CB                    ST2:
10CB    21 0000         LXI H,0
10CE    22 200F         SHLD LOPVAR
10D1    22 2009         SHLD STKGOS
10D4                    ST3:
10D4    3E3E            MVI A,">"
10D6    CD 155D         CALL GETLN
10D9    D5              PUSH D
10DA    CD 18A9         CALL DBUFF
10DD    CD 106C         CALL TSTNUM
10E0    CD 1015         CALL IGNB
10E3    7C              MOV A,H
```

Abb. 3.3-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 4
.MAIN. - RDK  B A S I C  INTERPRETER 3K  780524  V3.3

10E4    B5              ORA L
10E5    C1              POP B
10E6    CA 1876         JZ DIRECT
10E9    1B              DCX D
10EA    7C              MOV A,H
10EB    12              STAX D
10EC    1B              DCX D
10ED    7D              MOV A,L
10EE    12              STAX D
10EF    C5              PUSH B
10F0    D5              PUSH D
10F1    79              MOV A,C
10F2    93              SUB E
10F3    F5              PUSH PSW
```

```
10F4     CD 15A0       CALL FNDLN
10F7     D5            PUSH D
10F8     C2 110B       JNZ ST4
10FB     D5            PUSH D
10FC     CD 15BE       CALL FNDNXT
10FF     C1            POP B
1100     2A 201B       LHLD TXTUNF
1103     CD 1661       CALL MVUP
1106     60            MOV H,B
1107     69            MOV L,C
1108     22 201B       SHLD TXTUNF
110B                   ST4:
110B     C1            POP B
110C     2A 201B       LHLD TXTUNF
110F     F1            POP PSW
1110     E5            PUSH H
1111     FE03          CPI 3
1113     CA 10B8       JZ RSTART
1116     85            ADD L
1117     6F            MOV L,A
1118     3E00          MVI A,0
111A     8C            ADC H
111B     67            MOV H,A
111C     CD 18B1       CALL DTXTE
111F     CD 100F       CALL COMP
1122     D2 1556       JNC QSORRY
1125     22 201B       SHLD TXTUNF
1128     D1            POP D
1129     CD 166C       CALL MVDOWN
112C     D1            POP D
112D     E1            POP H
112E     CD 1661       CALL MVUP
1131     C3 10D4       JMP ST3
                       ;
                       ;************************************
                       ;  NEW  STOP  RUN  GOTO      UPR
                       ;************************************
                       ;
1134                   NEW:
1134     CD 1521       CALL ENDCHK
1137     21 214E       LXI H,TXTBGN
113A     22 201B       SHLD TXTUNF
113D                   STOP:
113D     CD 1521       CALL ENDCHK
1140     C3 10B8       JMP RSTART
1143                   RUN:
1143     CD 1521       CALL ENDCHK
```

Abb. 3.3-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                       PAGE 5
.MAIN. - RDK  B A S I C  INTERPRETER 3K  780524  V3.3


1146     11 214E       LXI D,TXTBGN
1149                   RUNNXL:
1149     21 0000       LXI H,0
114C     CD 15A8       CALL FNDLP
114F     DA 10B8       JC RSTART
1152                   RUNTSL:
```

Abb. 3.3-1

```
1152    EB              XCHG
1153    22 2007         SHLD CURRNT
1156    EB              XCHG
1157    13              INX D
1158    13              INX D
1159                    RUNSML:
1159    CD 1A5C         CALL CONT
115C    21 1772         LXI H,TAB2-1
115F    C3 1879         JMP EXEC
1162                    GOTO:
1162    CD 1363         CALL EXPR
1165    D5              PUSH D
1166    CD 1521         CALL ENDCHK
1169    CD 15A0         CALL FNDLN
116C    C2 1097         JNZ AHOW
116F    F1              POP PSW
1170    C3 1152         JMP RUNTSL
                        ;
                        ;****************************************
                        ;      LIST  PRINT          UPR
                        ;****************************************
                        ;
1173                    LIST:
1173    CD 106C         CALL TSTNUM
1176    CD 1521         CALL ENDCHK
1179    CD 15A0         CALL FNDLN
117C                    LS1:
117C    DA 10B8         JC RSTART
117F    CD 164C         CALL PRTLN
1182    CD 1A5C         CALL CONT
1185    CD 15A8         CALL FNDLP
1188    C3 117C         JMP LS1
118B                    PRINT:
118B    0E06            MVI C,6
118D    CD 1058         CALL TSTC
1190    3B              .BYTE 03BH
1191    06              .BYTE PR2-.-1
1192    CD 16F9         CALL CRLF
1195    C3 1159         JMP RUNSML
1198                    PR2:
1198    CD 1058         CALL TSTC
119B    0D              .BYTE 0DH
119C    06              .BYTE PR0-.-1
119D    CD 16F9         CALL CRLF
11A0    C3 1149         JMP RUNNXL
11A3                    PR0:
11A3    CD 1058         CALL TSTC
11A6    23              .BYTE 023H
11A7    07              .BYTE PR1-.-1
11A8    CD 1363         CALL EXPR
11AB    4D              MOV C,L
11AC    C3 11B5         JMP PR3
11AF                    PR1:
11AF    CD 15D8         CALL QTSTG
11B2    C3 11C6         JMP PR8
```

Abb. 3.3-1

```
11B5               PR3:
11B5   CD 1058     CALL TSTC
11B8   2C          .BYTE 02CH
11B9   06          .BYTE PR6-.-1
11BA   CD 150E     CALL FIN
11BD   C3 11A3     JMP PR0
11C0               PR6:
11C0   CD 16F9     CALL CRLF
11C3   CD 101D     CALL FINI
11C6               PR8:
11C6   CD 1363     CALL EXPR
11C9   C5          PUSH B
11CA   CD 1608     CALL PRTNUM
11CD   C1          POP B
11CE   C3 11B5     JMP PR3
                   ;***********************************
                   ;  GOSUB       RETURN      UPR
                   ;***********************************
                   ;
11D1               GOSUB:
11D1   CD 1697     CALL PUSHA
11D4   CD 1363     CALL EXPR
11D7   D5          PUSH D
11D8   CD 15A0     CALL FNDLN
11DB   C2 1097     JNZ AHOW
11DE   2A 2007     LHLD CURRNT
11E1   E5          PUSH H
11E2   2A 2009     LHLD STKGOS
11E5   E5          PUSH H
11E6   21 0000     LXI H,0
11E9   22 200F     SHLD LOPVAR
11EC   39          DAD SP
11ED   22 2009     SHLD STKGOS
11F0   C3 1152     JMP RUNTSL
11F3               RETURN:
11F3   CD 1521     CALL ENDCHK
11F6   2A 2009     LHLD STKGOS
11F9   7C          MOV A,H
11FA   B5          ORA L
11FB   CA 1527     JZ QWHAT
11FE   F9          SPHL
11FF   E1          POP H
1200   22 2009     SHLD STKGOS
1203   E1          POP H
1204   22 2007     SHLD CURRNT
1207   D1          POP D
1208   CD 167B     CALL POPA
120B   CD 101D     CALL FINI
                   ;***********************************
                   ;     FOR      NEXT           UPR
                   ;***********************************
                   ;
120E               FOR:
120E   CD 1697     CALL PUSHA
1211   CD 14F5     CALL SETVAL
1214   2B          DCX H
1215   22 200F     SHLD LOPVAR
1218   21 1846     LXI H,TAB5-1
121B   C3 1879     JMP EXEC
121E               FR1:
121E   CD 1363     CALL EXPR
```

Abb. 3.3-1

```
1221    22 2013     SHLD LOPLMT
1224    21 184E     LXI H,TAB6-1
1227    C3 1879     JMP EXEC
122A                FR2:
122A    CD 1363     CALL EXPR
122D    C3 1233     JMP FR4
1230                FR3:
1230    21 0001     LXI H,1
1233                FR4:
1233    22 2011     SHLD LOPINC
1236                FR5:
1236    2A 2007     LHLD CURRNT
1239    22 2015     SHLD LOPLN
123C    EB          XCHG
123D    22 2017     SHLD LOPPT
1240    01 000A     LXI B,0AH
1243    2A 200F     LHLD LOPVAR
1246    EB          XCHG
1247    60          MOV H,B
1248    68          MOV L,B
1249    39          DAD SP
124A    3E          .BYTE 03EH
124B                FR7:
124B    09          DAD B
124C    7E          MOV A,M
124D    23          INX H
124E    B6          ORA M
124F    CA 126C     JZ FR8
1252    7E          MOV A,M
1253    2B          DCX H
1254    BA          CMP D
1255    C2 124B     JNZ FR7
1258    7E          MOV A,M
1259    BB          CMP E
125A    C2 124B     JNZ FR7
125D    EB          XCHG
125E    21 0000     LXI H,0
1261    39          DAD SP
1262    44          MOV B,H
1263    4D          MOV C,L
1264    21 000A     LXI H,0AH
1267    19          DAD D
1268    CD 166C     CALL MVDOWN
126B    F9          SPHL
126C                FR8:
126C    2A 2017     LHLD LOPPT
126F    EB          XCHG
1270    CD 101D     CALL FINI
1273                NEXT:
1273    CD 1024     CALL TSTV
1276    DA 1527     JC QWHAT
1279    22 200B     SHLD VARNXT
127C                NX0:
127C    D5          PUSH D
127D    EB          XCHG
127E    2A 200F     LHLD LOPVAR
1281    7C          MOV A,H
1282    B5          ORA L
1283    CA 1528     JZ AWHAT
1286    CD 100F     CALL COMP
1289    CA 1296     JZ NX3
```

Abb. 3.3-1

```
128C    D1          POP D
128D    CD 167B     CALL POPA
1290    2A 200B     LHLD VARNXT
1293    C3 127C     JMP NX0
1296                NX3:
1296    5E          MOV E,M
1297    23          INX H
1298    56          MOV D,M
1299    2A 2011     LHLD LOPINC
129C    E5          PUSH H
129D    7C          MOV A,H
129E    AA          XRA D
129F    7A          MOV A,D
12A0    19          DAD D
12A1    FA 12A8     JM NX4
12A4    AC          XRA H
12A5    FA 12CC     JM NX5
12A8                NX4:
12A8    EB          XCHG
12A9    2A 200F     LHLD LOPVAR
12AC    73          MOV M,E
12AD    23          INX H
12AE    72          MOV M,D
12AF    2A 2013     LHLD LOPLMT
12B2    F1          POP PSW
12B3    B7          ORA A
12B4    F2 12B8     JP NX1
12B7    EB          XCHG
12B8                NX1:
12B8    CD 14EB     CALL CKHLDE
12BB    D1          POP D
12BC    DA 12CE     JC NX2
12BF    2A 2015     LHLD LOPLN
12C2    22 2007     SHLD CURRNT
12C5    2A 2017     LHLD LOPPT
12C8    EB          XCHG
12C9    CD 101D     CALL FINI
12CC                NX5:
12CC    E1          POP H
12CD    D1          POP D
12CE                NX2:
12CE    CD 167B     CALL POPA
12D1    CD 101D     CALL FINI
                    ;*************************************
                    ; REM  IF  INPUT  LET  UPR
                    ;*************************************
12D4                REM:
12D4    21 0000     LXI H,0
12D7    C3 12DD     JMP IFFR
12DA                IFF:
12DA    CD 1363     CALL EXPR
12DD                IFFR:
12DD    7C          MOV A,H
12DE    B5          ORA L
12DF    C2 1159     JNZ RUNSML
12E2    CD 15C0     CALL FNDSKP
12E5    D2 1152     JNC RUNTSL
12E8    C3 10B8     JMP RSTART
12EB                INPERR:
12EB    2A 200D     LHLD STKINP
12EE    F9          SPHL
```

Abb. 3.3-1

```
12EF    E1         POP H
12F0    22 2007    SHLD CURRNT
12F3    D1         POP D
12F4    D1         POP D
12F5               INPUT:
12F5               IP1:
12F5    D5         PUSH D
12F6    CD 15D8    CALL QTSTG
12F9    C3 1305    JMP IP2
12FC    CD 1024    CALL TSTV
12FF    DA 1343    JC IP4
1302    C3 1317    JMP IP3
1305               IP2:
1305    D5         PUSH D
1306    CD 1024    CALL TSTV
1309    DA 1527    JC QWHAT
130C    1A         LDAX D
130D    4F         MOV C,A
130E    97         SUB A
130F    12         STAX D
1310    D1         POP D
1311    CD 15CA    CALL PRTSTG
1314    79         MOV A,C
1315    1B         DCX D
1316    12         STAX D
1317               IP3:
1317    D5         PUSH D
1318    EB         XCHG
1319    2A 2007    LHLD CURRNT
131C    E5         PUSH H
131D    21 12F5    LXI H,IP1
1320    22 2007    SHLD CURRNT
1323    21 0000    LXI H,0
1326    39         DAD SP
1327    22 200D    SHLD STKINP
132A    D5         PUSH D
132B    3E3A       MVI A,03AH
132D    CD 155D    CALL GETLN
1330    CD 18A9    CALL DBUFF
1333    CD 1363    CALL EXPR
1336    CD 1A5C    CALL CONT
1339    D1         POP D
133A    EB         XCHG
133B    73         MOV M,E
133C    23         INX H
133D    72         MOV M,D
133E    E1         POP H
133F    22 2007    SHLD CURRNT
1342    D1         POP D
1343               IP4:
1343    F1         POP PSW
1344    CD 1058    CALL TSTC
1347    2C         .BYTE ","
1348    03         .BYTE IP5-.-1
1349    C3 12F5    JMP IP1
134C               IP5:
134C    CD 101D    CALL FINI
134F               DEFLT:
134F    1A         LDAX D
1350    FE0D       CPI 0DH
1352    CA 1360    JZ LT1
```

Abb. 3.3-1

```
1355                LET:
1355    CD 14F5     CALL SETVAL
1358    CD 1058     CALL TSTC
135B    2C          .BYTE ","
135C    03          .BYTE LT1-.-1
135D    C3 1355     JMP LET
1360                LT1:
1360    CD 101D     CALL FINI
                    ;************************************
                    ;     EXPR                UPR
                    ;************************************
1363                EXPR:
1363    CD 13AB     CALL EXPR2
1366    E5          PUSH H
1367                EXPR1:
1367    21 1858     LXI H,TAB8-1
136A    C3 1879     JMP EXEC
136D                XP11:
136D    CD 1396     CALL XP18
1370    D8          RC
1371    6F          MOV L,A
1372    C9          RET
1373                XP12:
1373    CD 1396     CALL XP18
1376    C8          RZ
1377    6F          MOV L,A
1378    C9          RET
1379                XP13:
1379    CD 1396     CALL XP18
137C    C8          RZ
137D    D8          RC
137E    6F          MOV L,A
137F    C9          RET
1380                XP14:
1380    CD 1396     CALL XP18
1383    6F          MOV L,A
1384    C8          RZ
1385    D8          RC
1386    6C          MOV L,H
1387    C9          RET
1388                XP15:
1388    CD 1396     CALL XP18
138B    C0          RNZ
138C    6F          MOV L,A
138D    C9          RET
138E                XP16:
138E    CD 1396     CALL XP18
1391    D0          RNC
1392    6F          MOV L,A
1393    C9          RET
1394                XP17:
1394    E1          POP H
1395    C9          RET
1396                XP18:
1396    79          MOV A,C
1397    E1          POP H
1398    C1          POP B
1399    E5          PUSH H
139A    C5          PUSH B
139B    4F          MOV C,A
```

Abb. 3.3-1

```
139C    CD 13AB     CALL EXPR2
139F    EB          XCHG
13A0    E3          XTHL
13A1    CD 14EB     CALL CKHLDE
13A4    D1          POP D
13A5    21 0000     LXI H,0
13A8    3E01        MVI A,1
13AA    C9          RET
13AB                EXPR2:
13AB    CD 1058     CALL TSTC
13AE    2D          .BYTE 02DH
13AF    06          .BYTE XP21-.-1
13B0    21 0000     LXI H,0
13B3    C3 13DD     JMP XP26
13B6                XP21:
13B6    CD 1058     CALL TSTC
13B9    2B          .BYTE 02BH
13BA    00          .BYTE XP22-.-1
13BB                XP22:
13BB    CD 13E7     CALL EXPR3
13BE                XP23:
13BE    CD 1058     CALL TSTC
13C1    2B          .BYTE 02BH
13C2    15          .BYTE XP25-.-1
13C3    E5          PUSH H
13C4    CD 13E7     CALL EXPR3
13C7                XP24:
13C7    EB          XCHG
13C8    E3          XTHL
13C9    7C          MOV A,H
13CA    AA          XRA D
13CB    7A          MOV A,D
13CC    19          DAD D
13CD    D1          POP D
13CE    FA 13BE     JM XP23
13D1    AC          XRA H
13D2    F2 13BE     JP XP23
13D5    C3 1096     JMP QHOW
13D8                XP25:
13D8    CD 1058     CALL TSTC
13DB    2D          .BYTE 02DH
13DC    92          .BYTE XP42-.-1
13DD                XP26:
13DD    E5          PUSH H
13DE    CD 13E7     CALL EXPR3
13E1    CD 14D6     CALL CHGSGN
13E4    C3 13C7     JMP XP24
13E7                EXPR3:
13E7    CD 144B     CALL EXPR4
13EA                XP31:
13EA    CD 1058     CALL TSTC
13ED    2A          .BYTE 02AH
13EE    2D          .BYTE XP34-.-1
13EF    E5          PUSH H
13F0    CD 144B     CALL EXPR4
13F3    0600        MVI B,0
13F5    CD 14D3     CALL CHKSGN
13F8    E3          XTHL
13F9    CD 14D3     CALL CHKSGN
13FC    EB          XCHG
13FD    E3          XTHL
```

Abb. 3.3-1

```
13FE   7C        MOV A,H
13FF   B7        ORA A
1400   CA 1409   JZ XP32
1403   7A        MOV A,D
1404   B2        ORA D
1405   EB        XCHG
1406   C2 1097   JNZ AHOW
1409             XP32:
1409   7D        MOV A,L
140A   21 0000   LXI H,0
140D   B7        ORA A
140E   CA 143D   JZ XP35
1411             XP33:
1411   19        DAD D
1412   DA 1097   JC AHOW
1415   3D        DCR A
1416   C2 1411   JNZ XP33
1419   C3 143D   JMP XP35
141C             XP34:
141C   CD 1058   CALL TSTC
141F   2F        .BYTE 02FH
1420   4E        .BYTE XP42-.-1
1421   E5        PUSH H
1422   CD 144B   CALL EXPR4
1425   0600      MVI B,0
1427   CD 14D3   CALL CHKSGN
142A   E3        XTHL
142B   CD 14D3   CALL CHKSGN
142E   EB        XCHG
142F   E3        XTHL
1430   EB        XCHG
1431   7A        MOV A,D
1432   B3        ORA E
1433   CA 1097   JZ AHOW
1436   C5        PUSH B
1437   CD 14B6   CALL DIVIDE
143A   60        MOV H,B
143B   69        MOV L,C
143C   C1        POP B
143D             XP35:
143D   D1        POP D
143E   7C        MOV A,H
143F   B7        ORA A
1440   FA 1096   JM QHOW
1443   78        MOV A,B
1444   B7        ORA A
1445   FC 14D6   CM CHGSGN
1448   C3 13EA   JMP XP31
144B             EXPR4:
144B   21 17FE   LXI H,TAB4-1
144E   C3 1879   JMP EXEC
1451             XP40:
1451   CD 1024   CALL TSTV
1454   DA 145C   JC XP41
1457   7E        MOV A,M
1458   23        INX H
1459   66        MOV H,M
145A   6F        MOV L,A
145B   C9        RET
145C             XP41:
145C   CD 106C   CALL TSTNUM
```

Abb. 3.3-1

```
145F    78          MOV A,B
1460    B7          ORA A
1461    C0          RNZ
1462                PARN:
1462    CD 1058     CALL TSTC
1465    28          .BYTE 028H
1466    09          .BYTE XP43-.-1
1467    CD 1363     CALL EXPR
146A    CD 1058     CALL TSTC
146D    29          .BYTE 029H
146E    01          .BYTE XP43-.-1
146F                XP42:
146F    C9          RET
1470                XP43:
1470    C3 1527     JMP QWHAT
1473                RND:
1473    CD 1462     CALL PARN
1476    7C          MOV A,H
1477    B7          ORA A
1478    FA 1096     JM QHOW
147B    B5          ORA L
147C    CA 1096     JZ QHOW
147F    D5          PUSH D
1480    E5          PUSH H
1481    2A 2019     LHLD RANPNT
1484    11 2005     LXI D,LSTROM
1487    CD 100F     CALL COMP
148A    DA 1490     JC RA1
148D    21 16C0     LXI H,START
1490                RA1:
1490    5E          MOV E,M
1491    23          INX H
1492    56          MOV D,M
1493    22 2019     SHLD RANPNT
1496    E1          POP H
1497    EB          XCHG
1498    C5          PUSH B
1499    CD 14B6     CALL DIVIDE
149C    C1          POP B
149D    D1          POP D
149E    23          INX H
149F    C9          RET
14A0                ABS:
14A0    CD 1462     CALL PARN
14A3    1B          DCX D
14A4    CD 14D3     CALL CHKSGN
14A7    13          INX D
14A8    C9          RET
14A9                SIZE:
14A9    2A 201B     LHLD TXTUNF
14AC    D5          PUSH D
14AD    EB          XCHG
14AE    2A 2148     LHLD TXTEND
14B1    CD 14CC     CALL SUBDE
14B4    D1          POP D
14B5    C9          RET
                    ;*************************************
                    ; DIVIDE  SUBDE  CHKSGN CHGSGN
                    ; CKHLDE         UPR
                    ;*************************************
                    ;
```

Abb. 3.3-1

```
14B6                DIVIDE:
14B6     E5         PUSH H
14B7     6C         MOV L,H
14B8     2600       MVI H,0
14BA     CD 14C1    CALL DV1
14BD     41         MOV B,C
14BE     7D         MOV A,L
14BF     E1         POP H
14C0     67         MOV H,A
14C1                DV1:
14C1     0EFF       MVI C,0FFH
14C3                DV2:
14C3     0C         INR C
14C4     CD 14CC    CALL SUBDE
14C7     D2 14C3    JNC DV2
14CA     19         DAD D
14CB     C9         RET
14CC                SUBDE:
14CC     7D         MOV A,L
14CD     93         SUB E
14CE     6F         MOV L,A
14CF     7C         MOV A,H
14D0     9A         SBB D
14D1     67         MOV H,A
14D2     C9         RET
14D3                CHKSGN:
14D3     7C         MOV A,H
14D4     B7         ORA A
14D5     F0         RP
14D6                CHGSGN:
14D6     7C         MOV A,H
14D7     B5         ORA L
14D8     C8         RZ
14D9     7C         MOV A,H
14DA     F5         PUSH PSW
14DB     2F         CMA
14DC     67         MOV H,A
14DD     7D         MOV A,L
14DE     2F         CMA
14DF     6F         MOV L,A
14E0     23         INX H
14E1     F1         POP PSW
14E2     AC         XRA H
14E3     F2 1096    JP QHOW
14E6     78         MOV A,B
14E7     EE80       XRI 080H
14E9     47         MOV B,A
14EA     C9         RET
14EB                CKHLDE:
14EB     7C         MOV A,H
14EC     AA         XRA D
14ED     F2 14F1    JP CK1
14F0     EB         XCHG
14F1                CK1:
14F1     CD 100F    CALL COMP
14F4     C9         RET
                    ;**************************************
                    ; SETVAL  FIN  ENDCHK   ERROR   UPR
                    ;**************************************
                    ;
14F5                SETVAL:
```

```
14F5    CD 1024     CALL TSTV
14F8    DA 1527     JC QWHAT
14FB    E5          PUSH H
14FC    CD 1058     CALL TSTC
14FF    3D          .BYTE 03DH
1500    0A          .BYTE SV1-.-1
1501    CD 1363     CALL EXPR
1504    44          MOV B,H
1505    4D          MOV C,L
1506    E1          POP H
1507    71          MOV M,C
1508    23          INX H
1509    70          MOV M,B
150A    C9          RET
150B                SV1:
150B    C3 1527     JMP QWHAT
150E                FIN:
150E    CD 1058     CALL TSTC
1511    3B          .BYTE 03BH
1512    04          .BYTE FI1-.-1
1513    F1          POP PSW
1514    C3 1159     JMP RUNSML
1517                FI1:
1517    CD 1058     CALL TSTC
151A    0D          .BYTE 0DH
151B    04          .BYTE FI2-.-1
151C    F1          POP PSW
151D    C3 1149     JMP RUNNXL
1520                FI2:
1520    C9          RET
1521                ENDCHK:
1521    CD 1015     CALL IGNB
1524    FE0D        CPI 0DH
1526    C8          RZ
1527                QWHAT:
1527    D5          PUSH D
1528                AWHAT:
1528    11 10AA     LXI D,WHAT
152B                ERROR:
152B    97          SUB A
152C    CD 15CA     CALL PRTSTG
152F    D1          POP D
1530    1A          LDAX D
1531    F5          PUSH PSW
1532    97          SUB A
1533    12          STAX D
1534    2A 2007     LHLD CURRNT
1537    E5          PUSH H
1538    7E          MOV A,M
1539    23          INX H
153A    B6          ORA M
153B    D1          POP D
153C    CA 10B8     JZ RSTART
153F    7E          MOV A,M
1540    B7          ORA A
1541    FA 12EB     JM INPERR
1544    CD 164C     CALL PRTLN
1547    1B          DCX D
1548    F1          POP PSW
1549    12          STAX D
154A    3E3F        MVI A,03FH
```

Abb. 3.3-1

Abb. 3.3-1

```
154C    CD 16FB     CALL OUTC
154F    97          SUB A
1550    CD 15CA     CALL PRTSTG
1553    C3 10B8     JMP RSTART
1556                QSORRY:
1556    D5          PUSH D
1557                ASORRY:
1557    11 10B1     LXI D,SORRY
155A    C3 152B     JMP ERROR
                    ;**********************************
                    ;  GETLN   FNDLN          UPR
                    ;**********************************
                    ;
155D                GETLN:
155D    CD 16FB     CALL OUTC
1560    CD 18A9     CALL DBUFF
1563                GL1:
1563    CD 1720     CALL CHKIO
1566    FE01        CPI 1
1568    CA 1588     JZ GL3
156B    CD 16FB     CALL OUTC
156E    FE0A        CPI 0AH
1570    CA 1563     JZ GL1
1573    B7          ORA A
1574    CA 1563     JZ GL1
1577    FE1B        CPI 01BH
1579    CA 1598     JZ GL4
157C    12          STAX D
157D    13          INX D
157E    FE0D        CPI 0DH
1580    C8          RZ
1581    7B          MOV A,E
1582    CD 18B9     CALL CXBUFE
1585    C2 1563     JNZ GL1
1588                GL3:
1588    7B          MOV A,E
1589    CD 18C0     CALL CXBUFA
158C    CA 1598     JZ GL4
158F    1B          DCX D
1590    3E08        MVI A,8
1592    CD 16FB     CALL OUTC
1595    C3 1563     JMP GL1
1598                GL4:
1598    CD 16F9     CALL CRLF
159B    3E0B        MVI A,0BH
159D    C3 155D     JMP GETLN
15A0                FNDLN:
15A0    7C          MOV A,H
15A1    B7          ORA A
15A2    FA 1096     JM QHOW
15A5    11 214E     LXI D,TXTBGN
15A8                FNDLP:
15A8                FL1:
15A8    E5          PUSH H
15A9    2A 201B     LHLD TXTUNF
15AC    2B          DCX H
15AD    CD 100F     CALL COMP
15B0    E1          POP H
15B1    D8          RC
15B2    1A          LDAX D
15B3    95          SUB L
```

Abb. 3.3-1

```
15B4    47          MOV B,A
15B5    13          INX D
15B6    1A          LDAX D
15B7    9C          SBB H
15B8    DA 15BF     JC FL2
15BB    1B          DCX D
15BC    B0          ORA B
15BD    C9          RET
15BE                FNDNXT:
15BE    13          INX D
15BF                FL2:
15BF    13          INX D
15C0                FNDSKP:
15C0    1A          LDAX D
15C1    FE0D        CPI 0DH
15C3    C2 15BF     JNZ FL2
15C6    13          INX D
15C7    C3 15A8     JMP FL1
                    ;**************************************
                    ;  PRTSTG  QTSTG  PRTNUM  PRTLN  UPR
                    ;**************************************
                    ;
15CA                PRTSTG:
15CA    47          MOV B,A
15CB                PS1:
15CB    1A          LDAX D
15CC    13          INX D
15CD    B8          CMP B
15CE    C8          RZ
15CF    CD 16FB     CALL OUTC
15D2    FE0D        CPI 0DH
15D4    C2 15CB     JNZ PS1
15D7    C9          RET
15D8                QTSTG:
15D8    CD 1058     CALL TSTC
15DB    22          .BYTE 022H
15DC    0F          .BYTE QT3-.-1
15DD    3E22        MVI A,022H
15DF                QT1:
15DF    CD 15CA     CALL PRTSTG
15E2    FE0D        CPI 0DH
15E4    E1          POP H
15E5    CA 1149     JZ RUNNXL
15E8                QT2:
15E8    23          INX H
15E9    23          INX H
15EA    23          INX H
15EB    E9          PCHL
15EC                QT3:
15EC    CD 1058     CALL TSTC
15EF    27          .BYTE 027H
15F0    05          .BYTE QT4-.-1
15F1    3E27        MVI A,027H
15F3    C3 15DF     JMP QT1
15F6                QT4:
15F6    CD 1058     CALL TSTC
15F9    5F          .BYTE 05FH
15FA    0C          .BYTE QT5-.-1
15FB    3E8D        MVI A,08DH
15FD    CD 16FB     CALL OUTC
1600    CD 16FB     CALL OUTC
```

Abb. 3.3-1

```
1603    E1          POP H
1604    C3 15E8     JMP QT2
1607                QT5:
1607    C9          RET
1608                PRTNUM:
1608    0600        MVI B,0
160A    CD 14D3     CALL CHKSGN
160D    F2 1613     JP PN1
1610    062D        MVI B,02DH
1612    0D          DCR C
1613                PN1:
1613    D5          PUSH D
1614    11 000A     LXI D,0AH
1617    D5          PUSH D
1618    0D          DCR C
1619    C5          PUSH B
161A                PN2:
161A    CD 14B6     CALL DIVIDE
161D    78          MOV A,B
161E    B1          ORA C
161F    CA 162A     JZ PN3
1622    E3          XTHL
1623    2D          DCR L
1624    E5          PUSH H
1625    60          MOV H,B
1626    69          MOV L,C
1627    C3 161A     JMP PN2
162A                PN3:
162A    C1          POP B
162B                PN4:
162B    0D          DCR C
162C    79          MOV A,C
162D    B7          ORA A
162E    FA 1639     JM PN5
1631    3E20        MVI A,020H
1633    CD 16FB     CALL OUTC
1636    C3 162B     JMP PN4
1639                PN5:
1639    78          MOV A,B
163A    B7          ORA A
163B    C4 16FB     CNZ OUTC
163E    5D          MOV E,L
163F                PN6:
163F    7B          MOV A,E
1640    FE0A        CPI 0AH
1642    D1          POP D
1643    C8          RZ
1644    C630        ADI 030H
1646    CD 16FB     CALL OUTC
1649    C3 163F     JMP PN6
164C                PRTLN:
164C    1A          LDAX D
164D    6F          MOV L,A
164E    13          INX D
164F    1A          LDAX D
1650    67          MOV H,A
1651    13          INX D
1652    0E04        MVI C,4
1654    CD 1608     CALL PRTNUM
1657    3E20        MVI A,020H
1659    CD 16FB     CALL OUTC
```

```
165C    97          SUB A
165D    CD 15CA     CALL PRTSTG
1660    C9          RET
                    ;***********************************
                    ; MVUP  MVDOWN  POPA  PUSHA      UPR
                    ;***********************************
                    ;
1661                MVUP:
1661    CD 100F     CALL COMP
1664    C8          RZ
1665    1A          LDAX D
1666    02          STAX B
1667    13          INX D
1668    03          INX B
1669    C3 1661     JMP MVUP
166C                MVDOWN:
166C    78          MOV A,B
166D    92          SUB D
166E    C2 1674     JNZ MD1
1671    79          MOV A,C
1672    93          SUB E
1673    C8          RZ
1674                MD1:
1674    1B          DCX D
1675    2B          DCX H
1676    1A          LDAX D
1677    77          MOV M,A
1678    C3 166C     JMP MVDOWN
167B                POPA:
167B    C1          POP B
167C    E1          POP H
167D    22 200F     SHLD LOPVAR
1680    7C          MOV A,H
1681    B5          ORA L
1682    CA 1695     JZ PP1
1685    E1          POP H
1686    22 2011     SHLD LOPINC
1689    E1          POP H
168A    22 2013     SHLD LOPLMT
168D    E1          POP H
168E    22 2015     SHLD LOPLN
1691    E1          POP H
1692    22 2017     SHLD LOPPT
1695                PP1:
1695    C5          PUSH B
1696    C9          RET
1697                PUSHA:
1697    21 2045     LXI H,STKLMT
169A    CD 14D6     CALL CHGSGN
169D    C1          POP B
169E    39          DAD SP
169F    D2 1556     JNC QSORRY
16A2    2A 200F     LHLD LOPVAR
16A5    7C          MOV A,H
16A6    B5          ORA L
16A7    CA 16BD     JZ PU1
16AA    2A 2017     LHLD LOPPT
16AD    E5          PUSH H
16AE    2A 2015     LHLD LOPLN
16B1    E5          PUSH H
16B2    2A 2013     LHLD LOPLMT
```

Abb. 3.3-1

Abb. 3.3-1

```
16B5    E5          PUSH H
16B6    2A 2011     LHLD LOPINC
16B9    E5          PUSH H
16BA    2A 200F     LHLD LOPVAR
16BD                PU1:
16BD    E5          PUSH H
16BE    C5          PUSH B
16BF    C9          RET
                    ;****************************
                    ;*  OUTC  CHKIO  UPR       *
                    ;****************************
                    ;

16C0                START:
16C0    31 210F     LXI SP,STACK
16C3    3EFF        MVI A,0FFH
16C5                INIT:
16C5    32 2006     STA OCSW
16C8    1603        MVI D,3
16CA                PATLOP:
16CA    CD 16F9     CALL CRLF
16CD    15          DCR D
16CE    C2 16CA     JNZ PATLOP
16D1    97          SUB A
16D2    11 173A     LXI D,MSG1
16D5    CD 15CA     CALL PRTSTG
16D8    21 16C0     LXI H,START
16DB    22 2019     SHLD RANPNT
16DE    21 214E     LXI H,TXTBGN
16E1    22 201B     SHLD TXTUNF
16E4    21 23A8     LXI H,TXTE
16E7    22 214B     SHLD TXTEND
16EA    21 23AA     LXI H,BUFA
16ED    22 214A     SHLD BUFFER
16F0    21 23EA     LXI H,BUFE
16F3    22 214C     SHLD BUFEND
16F6    C3 10B8     JMP RSTART

16F9                CRLF:
16F9    3E0D        MVI A,0DH
16FB                OUTC:
16FB    C5          PUSH B
16FC    F5          PUSH PSW
16FD    3A 2006     LDA OCSW
1700    B7          ORA A
1701                OC2:
1701    C2 1707     JNZ OC3
1704    F1          POP PSW
1705    C1          POP B
1706    C9          RET
1707                OC3:
1707    F1          POP PSW
1708    F5          PUSH PSW
1709    4F          MOV C,A
170A                LPT:
170A    79          MOV A,C
170B    FE0D        CPI 0DH
170D    CA 1716     JZ LINEF
1710                H1:
1710    CD 1009     CALL ECHO
```

Abb. 3.3-1

```
1713    F1            POP PSW
1714    C1            POP B
1715    C9            RET
1716                  LINEF:
1716    0E0D          MVI C,0DH
1718    CD 1009       CALL ECHO
171B    0E0A          MVI C,0AH
171D    C3 1710       JMP H1
1720                  CHKIO:
1720    CD 1006       CALL CI
1723    E67F          ANI 7FH
1725    FE02          CPI 2
1727    C2 1734       JNZ CI1
172A    3A 2006       LDA OCSW
172D    2F            CMA
172E    32 2006       STA OCSW
1731    C3 1720       JMP CHKIO
1734                  CI1:
1734    FE03          CPI 3
1736    C0            RNZ
1737    C3 10B8       JMP RSTART
173A                  MSG1:
173A    52444B205052  .ASCII "RDK PRO\
1740    4F4D505420    \MPT "
1745    424153494320  .ASCII "BASIC V\
174B    56332E322033  \3.2 3K\
1751    4B            \
1752    0D0A          "
                      ;**********************************
                      ;* TABLES DIRECT EXEC             *
                      ;**********************************
                      .OPSYN .WORD,DWA;
                      .OPSYN .ASCIZ,TX
                      ;
                      ;
1754                  TAB1:
1754    4C49535400    TX "LIST"
1759    1173          DWA LIST
175B    52554E00      TX "RUN"
175F    1143          DWA RUN
1761    4E455700      TX "NEW"
1765    1134          DWA NEW
1767    42594500      TX "BYE"
176B    18F8          DWA BYE
176D    454E4400      TX "END"
1771    18C7          DWA END
1773                  TAB2:
1773    4E45585400    TX "NEXT"
1778    1273          DWA NEXT
177A    4C455400      TX "LET"
177E    1355          DWA LET
1780    494600        TX "IF"
1783    12DA          DWA IFF
1785    474F544F00    TX "GOTO"
178A    1162          DWA GOTO
178C    474F53554200  TX "GOSUB"
1792    11D1          DWA GOSUB
1794    52455455524E  TX "RETURN"\
179A    00            \
179B    11F3          DWA RETURN
179D    52454D00      TX "REM"
```

Abb. 3.3-1

```
17A1    12D4            DWA REM
17A3    464F5200        TX "FOR"
17A7    120E            DWA FOR
17A9    494E50555400    TX "INPUT"
17AF    12F5            DWA INPUT
17B1    5052494E5400    TX "PRINT"
17B7    118B            DWA PRINT
17B9    53544F5000      TX "STOP"
17BE    113D            DWA STOP
17C0    43414C4C00      TX "CALL"
17C5    18FC            DWA CALL
17C7    4F5554434841    TX "OUTCHAR"\
17CD    5200            \
17CF    1A01            DWA OUTCHAR
17D1    4F555400        TX "OUT"
17D5    1909            DWA OUT
17D7    4F2400          TX "O$"
17DA    195A            DWA O
17DC    492400          TX "I$"
17DF    1967            DWA I
17E1    504F4B4500      TX "POKE"
17E6    19A0            DWA POKE
17E8    54414200        TX "TAB"
17EC    192F            DWA TAB
17EE    4259544500      TX "BYTE"
17F3    19B5            DWA BYTE
17F5    574F524400      TX "WORD"
17FA    198F            DWA WORD
17FC    00              .BYTE 0
17FD    134F            DWA DEFLT
17FF                    TAB4:
17FF    524E4400        TX "RND"
1803    1473            DWA RND
1805    41425300        TX "ABS"
1809    14A0            DWA ABS
180B    53495A4500      TX "SIZE"
1810    14A9            DWA SIZE
1812    5045454B00      TX "PEEK"
1817    1999            DWA PEEK
1819    494E43484152    TX "INCHAR"\
181F    00              \
1820    1A0B            DWA INCHAR
1822    48455800        TX "HEX"
1826    1A12            DWA HEX
1828    494E00          TX "IN"
182B    1940            DWA IN
182D    2700            TX "'"
182F    19E9            DWA QUOTE
1831    544F5000        TX "TOP"
1835    19F7            DWA TOP
1837    4C454E00        TX "LEN"
183B    19FC            DWA LENGTH
183D    4353545300      TX "CSTS"
1842    19E1            DWA CSTAT
1844    00              .BYTE 0
1845    1451            DWA XP40
1847                    TAB5:
1847    544F00          TX "TO"
184A    121E            DWA FR1
184C    00              .BYTE 0
184D    1527            DWA QWHAT
```

Abb. 3.3-1

```
184F                  TAB6:
184F    5354455000    TX "STEP"
1854    122A          DWA FR2
1856    00            .BYTE 0
1857    1230          DWA FR3
1859                  TAB8:
1859    3E3D00        TX ">="
185C    136D          DWA XP11
185E    2300          TX "#"
1860    1373          DWA XP12
1862    3E00          TX ">"
1864    1379          DWA XP13
1866    3D00          TX "="
1868    1388          DWA XP15
186A    3C3D00        TX "<="
186D    1380          DWA XP14
186F    3C00          TX "<"
1871    138E          DWA XP16
1873    00            .BYTE 0
1874    1394          DWA XP17
                      ;
                      ;
                      ;****************
                      ;* DIRECT MODUL *
                      ;****************
                      ;
                      ;
                      ;
1876                  DIRECT:
1876    21 1753       LXI H,TAB1-1
1879                  EXEC:
1879                  EX0:
1879    CD 1015       CALL IGNB
187C    D5            PUSH D
187D                  EX1:
187D    1A            LDAX D
187E    13            INX D
187F    FE2E          CPI "."
1881    CA 189B       JZ EX3
1884    23            INX H
1885    BE            CMP M
1886    CA 187D       JZ EX1
1889    3E00          MVI A,0
188B    1B            DCX D
188C    BE            CMP M
188D    CA 18A2       JZ EX5
1890                  EX2:
1890    23            INX H
1891    BE            CMP M
1892    C2 1890       JNZ EX2
1895    23            INX H
1896    23            INX H
1897    D1            POP D
1898    C3 1879       JMP EX0
189B                  EX3:
189B    3E00          MVI A,0
189D                  EX4:
189D    23            INX H
189E    BE            CMP M
189F    C2 189D       JNZ EX4
18A2                  EX5:
```

Abb. 3.3-1

```
18A2    23          INX H
18A3    7E          MOV A,M
18A4    23          INX H
18A5    66          MOV H,M
18A6    6F          MOV L,A
18A7    F1          POP PSW
18A8    E9          PCHL
                    ;***************
                    ;* END EXEC    *
                    ;***************
                    ;
                    ;
                    ;
18A9                DBUFF:
18A9    E5          PUSH H
18AA    2A 214A     LHLD BUFFER
18AD    54          MOV D,H
18AE    5D          MOV E,L
18AF    E1          POP H
18B0    C9          RET
                    ;
18B1                DTXTE:
18B1    E5          PUSH H
18B2    2A 2148     LHLD TXTEND
18B5    54          MOV D,H
18B6    5D          MOV E,L
18B7    E1          POP H
18B8    C9          RET
                    ;
18B9                CXBUFE:
18B9    E5          PUSH H
18BA    2A 214C     LHLD BUFEND
18BD    BD          CMP L
18BE    E1          POP H
18BF    C9          RET
                    ;
18C0                CXBUFA:
18C0    E5          PUSH H
18C1    2A 214A     LHLD BUFFER
18C4    BD          CMP L
18C5    E1          POP H
18C6    C9          RET
                    ;
18C7                END:
18C7    CD 1363     CALL EXPR
18CA    EB          XCHG
18CB    21 23A8     LXI H,TXTE
18CE    EB          XCHG
18CF    CD 100F     CALL COMP
18D2    DA 1557     JC ASORRY
18D5    7C          MOV A,H
18D6    B7          ORA A
18D7    FA 1557     JM ASORRY
18DA    7E          MOV A,M
18DB    2F          CMA
18DC    77          MOV M,A
18DD    46          MOV B,M
18DE    B8          CMP B
18DF    C2 1557     JNZ ASORRY
18E2    22 214C     SHLD BUFEND
18E5    7D          MOV A,L
```

Abb. 3.3-1

```
18E6    D684        SUI 132
18E8    6F          MOV L,A
18E9    7C          MOV A,H
18EA    DE00        SBI 0
18EC    67          MOV H,A
18ED    22 214A     SHLD BUFFER
18F0    2B          DCX H
18F1    2B          DCX H
18F2    22 2148     SHLD TXTEND
18F5    C3 10B8     JMP RSTART
                    ;
18F8                BYE:
18F8    FF          RST 7
18F9    C3 10B8     JMP RSTART
                    ;
                    ;
18FC                CALL:
18FC    CD 1363     CALL EXPR
18FF    D5          PUSH D
1900    01 1905     LXI B,HERE
1903    C5          PUSH B
1904    E9          PCHL
1905                HERE:
1905    D1          POP D
1906    CD 101D     CALL FINI
                    ;
1909                OUT:
1909    CD 1462     CALL PARN
190C    E5          PUSH H
190D    CD 1058     CALL TSTC
1910    3D          .BYTE "="
1911    1A          .BYTE RSV0-.-1
1912    CD 1363     CALL EXPR
1915    45          MOV B,L
1916    3ED3        MVI A,0D3H
1918    32 2002     STA IOBUFA
191B    E1          POP H
191C    7D          MOV A,L
191D    32 2003     STA IOBUFB
1920    3EC9        MVI A,0C9H
1922    32 2004     STA IOBUFC
1925    78          MOV A,B
1926    CD 2002     CALL IOBUFA
1929    CD 101D     CALL FINI
192C                RSV0:
192C    C3 1527     JMP QWHAT
                    ;
192F                TAB:
192F    CD 1462     CALL PARN
1932                A1:
1932    7C          MOV A,H
1933    B5          ORA L
1934    CC 101D     CZ FINI
1937    2B          DCX H
1938    3E20        MVI A,20H
193A    CD 16FB     CALL OUTC
193D    C3 1932     JMP A1
                    ;
1940                IN:
1940    CD 1462     CALL PARN
1943    E5          PUSH H
```

Abb. 3.3-1

```
1944    3EDB        MVI A,0DBH
1946    32 2002     STA IOBUFA
1949    E1          POP H
194A    7D          MOV A,L
194B    32 2003     STA IOBUFB
194E    3EC9        MVI A,0C9H
1950    32 2004     STA IOBUFC
1953    CD 2002     CALL IOBUFA
1956    2600        MVI H,0
1958    6F          MOV L,A
1959    C9          RET
                    ;
                    ;
195A                O:
195A    CD 1363     CALL EXPR
195D    D5          PUSH D
195E    EB          XCHG
195F    AF          XRA A
1960    CD 15CA     CALL PRTSTG
1963    D1          POP D
1964    CD 101D     CALL FINI
                    ;
                    ;
1967                I:
1967    CD 1363     CALL EXPR
196A    D5          PUSH D
196B    EB          XCHG
196C    2A 201B     LHLD TXTUNF
196F    EB          XCHG
1970    CD 100F     CALL COMP
1973    DA 1557     JC ASORRY
1976    CD 18A9     CALL DBUFF
1979    CD 1563     CALL GL1
197C    44          MOV B,H
197D    4D          MOV C,L
197E    EB          XCHG
197F    2B          DCX H
1980    CD 18A9     CALL DBUFF
1983    D5          PUSH D
1984    CD 1661     CALL MVUP
1987    AF          XRA A
1988    02          STAX B
1989    D1          POP D
198A    23          INX H
198B    CD 14CC     CALL SUBDE
198E    EB          XCHG
198F    21 2000     LXI H,LEGT
1992    73          MOV M,E
1993    23          INX H
1994    72          MOV M,D
1995    D1          POP D
1996    CD 101D     CALL FINI
                    ;
                    ;
1999                PEEK:
1999    CD 1462     CALL PARN
199C    6E          MOV L,M
199D    2600        MVI H,0
199F    C9          RET
                    ;
                    ;
```

Abb. 3.3-1

```
19A0                POKE:
19A0   CD 1363      CALL EXPR
19A3   E5           PUSH H
19A4   CD 1058      CALL TSTC
19A7   2C           .BYTE ","
19A8   09           .BYTE PK1-.-1
19A9   CD 1363      CALL EXPR
19AC   7D           MOV A,L
19AD   E1           POP H
19AE   77           MOV M,A
19AF   CD 101D      CALL FINI
19B2                PK1:
19B2   C3 1527      JMP QWHAT
                    ;
19B5                BYTE:
19B5   CD 1462      CALL PARN
19B8   7D           MOV A,L
19B9   CD 19CD      CALL WRIT2
19BC   CD 101D      CALL FINI
                    ;
19BF                WORD:
19BF   CD 1462      CALL PARN
19C2   7C           MOV A,H
19C3   CD 19CD      CALL WRIT2
19C6   7D           MOV A,L
19C7   CD 19CD      CALL WRIT2
19CA   CD 101D      CALL FINI
19CD                WRIT2:
19CD   F5           PUSH PSW
19CE   0F           RRC
19CF   0F           RRC
19D0   0F           RRC
19D1   0F           RRC
19D2   CD 19D6      CALL IST
19D5   F1           POP PSW
19D6                IST:
19D6   E60F         ANI 0FH
19D8   C690         ADI 90H
19DA   27           DAA
19DB   CE40         ACI 40H
19DD   27           DAA
19DE   C3 16FB      JMP OUTC
                    ;
19E1                CSTAT:
19E1   CD 100C      CALL CSTS
19E4   2F           CMA
19E5   6F           MOV L,A
19E6   2600         MVI H,0
19E8   C9           RET
                    ;
19E9                QUOTE:
19E9   1A           LDAX D
19EA   13           INX D
19EB   6F           MOV L,A
19EC   2600         MVI H,0
19EE   CD 1058      CALL TSTC
19F1   27           .BYTE "'"
19F2   01           .BYTE ASCI-.-1
19F3   C9           RET
19F4                ASCI:
19F4   C3 1527      JMP QWHAT
```

Abb. 3.3-1

```
                            ;
19F7                        TOP:
19F7        2A 201B         LHLD TXTUNF
19FA        23              INX H
19FB        C9              RET
                            ;
19FC                        LENGTH:
19FC        2A 2000         LHLD LEGT
19FF        2B              DCX H
1A00        C9              RET
                            ;
1A01                        OUTCHAR:
1A01        CD 1363         CALL EXPR
1A04        7D              MOV A,L
1A05        CD 16FB         CALL OUTC
1A08        CD 101D         CALL FINI
                            ;
1A0B                        INCHAR:
1A0B        CD 1720         CALL CHKIO
1A0E        2600            MVI H,0
1A10        6F              MOV L,A
1A11        C9              RET
                            ;
1A12                        HEX:
1A12        C5              PUSH B
1A13        21 0000         LXI H,0
1A16        CD 1058         CALL TSTC
1A19        28              .BYTE "("
1A1A        1D              .BYTE HN2-.-1
1A1B                        HNXTH:
1A1B        1A              LDAX D
1A1C        FE0D            CPI 0DH
1A1E        CA 1527         JZ QWHAT
1A21        CD 1A3D         CALL CNVBN
1A24        29              DAD H
1A25        29              DAD H
1A26        29              DAD H
1A27        29              DAD H
1A28        0600            MVI B,0
1A2A        4F              MOV C,A
1A2B        09              DAD B
1A2C        13              INX D
1A2D        CD 1058         CALL TSTC
1A30        29              .BYTE ")"
1A31        03              .BYTE HN1-.-1
1A32        C3 1A3B         JMP POPRET
1A35                        HN1:
1A35        C3 1A1B         JMP HNXTH
1A38                        HN2:
1A38        C3 1527         JMP QWHAT
1A3B                        POPRET:
1A3B        C1              POP B
1A3C        C9              RET
                            ;
                            ;
1A3D                        CNVBN:
1A3D        FE30            CPI 30H
1A3F        FA 1527         JM QWHAT
1A42        FE39            CPI 39H
1A44        FA 1A54         JM CONTC
1A47        CA 1A54         JZ CONTC
```

Abb. 3.3-1

```
1A4A    FE41        CPI 41H
1A4C    FA 1527     JM QWHAT
1A4F    FE47        CPI 47H
1A51    F2 1527     JP QWHAT
1A54                CONTC:
1A54    D630        SUI 30H
1A56    FE0A        CPI 10
1A58    F8          RM
1A59    D607        SUI 7
1A5B    C9          RET
                    ;
                    ;
                    ;
1A5C                CONT:
1A5C    CD 100C     CALL CSTS
1A5F    C8          RZ
1A60    CD 1006     CALL CI
1A63    FE03        CPI 3H
1A65    C0          RNZ
1A66    C3 10B8     JMP RSTART
                    ;
                    ;
1A69    00          ENDE:NOP
                    ;
2000                RAM=\"RAM LOCATION "
2000                .LOC RAM
                    ;
                    .OPSYN .BLKB,DS
                    ;
2000                LEGT:DS 2
2002                IOBUFA:DS 1
2003                IOBUFB:DS 1
2004                IOBUFC:DS 1
2005                LSTROM:DS 1
2006                OCSW:DS 1
2007                CURRNT:DS 2
2009                STKGOS:DS 2
200B                VARNXT:DS 2
200D                STKINP:DS 2
200F                LOPVAR:DS 2
2011                LOPINC:DS 2
2013                LOPLMT:DS 2
2015                LOPLN:DS 2
2017                LOPPT:DS 2
2019                RANPNT:DS 2
201B                TXTUNF:DS 2
201D                DS 40
2045                STKLMT: DS 2
2047                DS 200
210F                STACK: DS 2
2111                VARBGN:DS 55
2148                TXTEND:DS 2
214A                BUFFER:DS 2
214C                BUFEND:DS 2
214E                TXTBGN:DS 2
2150                DS 600
23A8                TXTE:DS 2
23AA                BUFA:DS 64
23EA                BUFE:DS 1
                    .END
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                       PAGE 30
.MAIN. - RDK  B A S I C  INTERPRETER 3K 780524  V3.3
+++++ SYMBOL TABLE +++++


A1      1932        ABS     14A0        AHOW    1097
ANBAS   1000        ANF     0FEE        ASCI    19F4
ASORRY  1557        AWHAT   1528        BEGINN  0FE3
BUFA    23AA        BUFE    23EA        BUFEND  214C
BUFFER  214A        BYE     18F8        BYTE    19B5
CALL    18FC        CHGSGN  14D6        CHKIO   1720
CHKSGN  14D3        CI      1006        CI1     1734
CK1     14F1        CKHLDE  14EB        CNVBN   1A3D
COMP    100F        CONT    1A5C        CONTC   1A54
CRLF    16F9        CSTAT   19E1        CSTS    100C
CURRNT  2007        CXBUFA  18C0        CXBUFE  18B9
DBUFF   18A9        DEFLT   134F        DIRECT  1876
DIVIDE  14B6        DTXTE   18B1        DV1     14C1
DV2     14C3        ECHO    1009        END     18C7
ENDCHK  1521        ENDE    1A69        ERROR   152B
EX0     1879        EX1     187D        EX2     1890
EX3     189B        EX4     189D        EX5     18A2
EXEC    1879        EXPR    1363        EXPR1   1367
EXPR2   13AB        EXPR3   13E7        EXPR4   144B
FI1     1517        FI2     1520        FIN     150E
FINI    101D        FL1     15A8        FL2     15BF
FNDLN   15A0        FNDLP   15A8        FNDNXT  15BE
FNDSKP  15C0        FOR     120E        FR1     121E
FR2     122A        FR3     1230        FR4     1233
FR5     1236        FR7     124B        FR8     126C
GETLN   155D        GL1     1563        GL3     1588
GL4     1598        GOSUB   11D1        GOTO    1162
H1      1710        HAUPTP  1000        HERE    1905
HEX     1A12        HN1     1A35        HN2     1A38
HNXTH   1A1B        HOW     109D        I       1967
IFF     12DA        IFFR    12DD        IGNB    1015
IN      1940        INCHAR  1A0B        INIT    16C5
INPERR  12EB        INPUT   12F5        IOBUFA  2002
IOBUFB  2003        IOBUFC  2004        IP1     12F5
IP2     1305        IP3     1317        IP4     1343
IP5     134C        IST     19D6        LEGT    2000
LENGTH  19FC        LET     1355        LINEF   1716
LIST    1173        LOPINC  2011        LOPLMT  2013
LOPLN   2015        LOPPT   2017        LOPVAR  200F
LPT     170A        LS1     117C        LSTROM  2005
LT1     1360        MD1     1674        MSG1    173A
MVDOWN  166C        MVUP    1661        NEW     1134
NEXT    1273        NX0     127C        NX1     12B8
NX2     12CE        NX3     1296        NX4     12A8
NX5     12CC        O       195A        OC2     1701
OC3     1707        OCSW    2006        OK      10A3
OUT     1909        OUTC    16FB        OUTCHA  1A01
PARN    1462        PATLOP  16CA        PEEK    1999
PK1     19B2        PN1     1613        PN2     161A
PN3     162A        PN4     162B        PN5     1639
PN6     163F        POKE    19A0        POPA    167B
POPRET  1A3B        PP1     1695        PR0     11A3
PR1     11AF        PR2     1198        PR3     11B5
PR6     11C0        PR8     11C6        PRINT   118B
PRTLN   164C        PRTNUM  1608        PRTSTG  15CA
PS1     15CB        PU1     16BD        PUSHA   1697
QHOW    1096        QSORRY  1556        QT1     15DF
QT2     15E8        QT3     15EC        QT4     15F6
QT5     1607        QTSTG   15D8        QUOTE   19E9
QWHAT   1527        RA1     1490        RAM     2000
RANPNT  2019        REM     12D4        RETURN  11F3
```

Abb. 3.3-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                            PAGE 31
.MAIN. - RDK  B A S I C  INTERPRETER 3K  780524  V3.3
+++++ SYMBOL TABLE +++++

RND     1473              RSTART 10B8              RSV0    192C
RUN     1143              RUNNXL 1149              RUNSML  1159
RUNTSL  1152              SETVAL 14F5              SIZE    14A9
SORRY   10B1              ST1    10BB              ST2     10CB
ST3     10D4              ST4    110B              STACK   210F
START   16C0              STKGOS 2009              STKINP  200D
STKLMT  2045              STOP   113D              SUBDE   14CC
SV1     150B              TAB    192F              TAB1    1754
TAB2    1773              TAB4   17FF              TAB5    1847
TAB6    184F              TAB8   1859              TC1     105D
TC2     1068              TN1    1073              TOP     19F7
TSTC    1058              TSTNUM 106C              TSTV    1024
TV1     1048              TXTBGN 214E              TXTE    23A8
TXTEND  2148              TXTUNF 201B              VARBGN  2111
VARNXT  200B              WHAT   10AA              WORD    19BF
WRIT2   19CD              XP11   136D              XP12    1373
XP13    1379              XP14   1380              XP15    1388
XP16    138E              XP17   1394              XP18    1396
XP21    13B6              XP22   13BB              XP23    13BE
XP24    13C7              XP25   13D8              XP26    13DD
XP31    13EA              XP32   1409              XP33    1411
XP34    141C              XP35   143D              XP40    1451
XP41    145C              XP42   146F              XP43    1470
.BLNK.  0000:03  X        .DATA.  0000*     X      .PROG.  0000'    X
```

Abb. 3.3-1

Abb. 3.3-2 Hex-Listing des RDK-BASIC

```
1000 C3 C0 16 C3 B8 10 C3 03 F0 C3 09 F0 C3 12 F0 7C ...............¦
1010 BA C0 7D BB C9 1A FE 20 C0 13 C3 15 10 F1 CD 0E ..}.... ........
1020 15 C3 27 15 CD 15 10 D6 40 D8 C2 48 10 13 CD 62 ..'.....@..H...b
1030 14 29 DA 96 10 D5 EB CD A9 14 CD 0F 10 DA 57 15 .)............W.
1040 2A 48 21 CD CC 14 D1 C9 FE 1B 3F D8 13 21 11 21 *H!.......?..!.!
1050 07 85 6F 3E 00 8C 67 C9 E3 CD 15 10 BE 23 CA 68 ..o>..g......#.h
1060 10 C5 4E 06 00 09 C1 1B 13 23 E3 C9 21 00 00 44 ..N......#..!..D
1070 CD 15 10 FE 30 D8 FE 3A D0 3E F0 A4 C2 96 10 04 ....0..:.>......
1080 C5 44 4D 29 29 09 29 1A 13 E6 0F 85 6F 3E 00 8C .DM)).).....o>..
1090 67 C1 1A F2 73 10 D5 11 9D 10 C3 2B 15 48 4F 57 g...s......+.HOW
10A0 3F 0D 0A 52 45 41 44 59 0D 0A 57 48 41 54 3F 0D ?..READY..WHAT?.
10B0 0A 53 4F 52 52 59 0D 0A 31 0F 21 CD F9 16 11 A3 .SORRY..1.!.....
10C0 10 97 CD CA 15 21 CC 10 22 07 20 21 00 00 22 0F .....!..". !..".
10D0 20 22 09 20 3E 3E CD 5D 15 D5 CD A9 18 CD 6C 10  ". >>.].......l.
10E0 CD 15 10 7C B5 C1 CA 76 18 1B 7C 12 1B 7D 12 C5 ...¦...v..¦..}..
10F0 D5 79 93 F5 CD A0 15 D5 C2 0B 11 D5 CD BE 15 C1 .y..............
1100 2A 1B 20 CD 61 16 60 69 22 1B 20 C1 2A 1B 20 F1 *. .a.`i". .*. .
1110 E5 FE 03 CA B8 10 85 6F 3E 00 8C 67 CD B1 18 CD .......o>..g....
1120 0F 10 D2 56 15 22 1B 20 D1 CD 6C 16 D1 E1 CD 61 ...V.". ..l....a
1130 16 C3 D4 10 CD 21 15 21 4E 21 22 1B 20 CD 21 15 .....!.!N!". .!.
1140 C3 B8 10 CD 21 15 11 4E 21 21 00 00 CD A8 15 DA ....!..N!!......
1150 B8 10 EB 22 07 20 EB 13 13 CD 5C 1A 21 72 17 C3 ...". .....\.!r..
1160 79 18 CD 63 13 D5 CD 21 15 CD A0 15 C2 97 10 F1 y..c...!........
1170 C3 52 11 CD 6C 10 CD 21 15 CD A0 15 DA B8 10 CD .R..l..!........
1180 4C 16 CD 5C 1A CD A8 15 C3 7C 11 0E 06 CD 58 10 L..\.....¦....X.
1190 3B 06 CD F9 16 C3 59 11 CD 58 10 0D 06 CD F9 16 ;.....Y..X......
```

```
11A0 C3 49 11 CD 58 10 23 07 CD 63 13 4D C3 B5 11 CD .I..X.#..c.M....
11B0 D8 15 C3 C6 11 CD 58 10 2C 06 CD 0E 15 C3 A3 11 .......X.,......
11C0 CD F9 16 CD 1D 10 CD 63 13 C5 CD 08 16 C1 C3 B5 .......c........
11D0 11 CD 97 16 CD 63 13 D5 CD A0 15 C2 97 10 2A 07 .....c.........*.
11E0 20 E5 2A 09 20 E5 21 00 00 22 0F 20 39 22 09 20  .*. .!..". 9".
11F0 C3 52 11 CD 21 15 2A 09 20 7C B5 CA 27 15 F9 E1 .R..!.*. |..'...
1200 22 09 20 E1 22 07 20 D1 CD 7B 16 CD 1D 10 CD 97 ". .". ..{......
1210 16 CD F5 14 2B 22 0F 20 21 46 18 C3 79 18 CD 63 ....+". !F..y..c
1220 13 22 13 20 21 4E 18 C3 79 18 CD 63 13 C3 33 12 .". !N..y..c..3.
1230 21 01 00 22 11 20 2A 07 20 22 15 20 EB 22 17 20 !..". *. ". .".
1240 01 0A 00 2A 0F 20 EB 60 68 39 3E 09 7E 23 B6 CA ...*. .`h9>.~#..
1250 6C 12 7E 2B BA C2 4B 12 7E BB C2 4B 12 EB 21 00 l.~+..K.~..K..!.
1260 00 39 44 4D 21 0A 00 19 CD 6C 16 F9 2A 17 20 EB .9DM!....l..*. .
1270 CD 1D 10 CD 24 10 DA 27 15 22 0B 20 D5 EB 2A 0F ....$..'.". ..*.
1280 20 7C B5 CA 28 15 CD 0F 10 CA 96 12 D1 CD 7B 16  |..(..........{.
1290 2A 0B 20 C3 7C 12 5E 23 56 2A 11 20 E5 7C AA 7A *. .|.^#V*. .|.z
12A0 19 FA A8 12 AC FA CC 12 EB 2A 0F 20 73 23 72 2A ..........*. s#r*
12B0 13 20 F1 B7 F2 B8 12 EB CD EB 14 D1 DA CE 12 2A . ..............*
12C0 15 20 22 07 20 2A 17 20 EB CD 1D 10 E1 D1 CD 7B . ". *. ........{
12D0 16 CD 1D 10 21 00 00 C3 DD 12 CD 63 13 7C B5 C2 ....!......c.|..
12E0 59 11 CD C0 15 D2 52 11 C3 B8 10 2A 0D 20 F9 E1 Y.....R....*. ..
12F0 22 07 20 D1 D1 D5 CD D8 15 C3 05 13 CD 24 10 DA ". ...........$..
1300 43 13 C3 17 13 D5 CD 24 10 DA 27 15 1A 4F 97 12 C......$..'..O..
1310 D1 CD CA 15 79 1B 12 D5 EB 2A 07 20 E5 21 F5 12 ....y....*. .!..
1320 22 07 20 21 00 00 39 22 0D 20 D5 3E 3A CD 5D 15 ". !..9". .>:.].
1330 CD A9 18 CD 63 13 CD 5C 1A D1 EB 73 23 72 E1 22 ....c..\...s#r."
1340 07 20 D1 F1 CD 58 10 2C 03 C3 F5 12 CD 1D 10 1A . ...X.,.........
1350 FE 0D CA 60 13 CD F5 14 CD 58 10 2C 03 C3 55 13 ...`.....X.,..U.
1360 CD 1D 10 CD AB 13 E5 21 58 18 C3 79 18 CD 96 13 .......!X..y....
1370 D8 6F C9 CD 96 13 C8 6F C9 CD 96 13 C8 D8 6F C9 .o......o......o.
1380 CD 96 13 6F C8 D8 6C C9 CD 96 13 C0 6F C9 CD 96 ...o..l.....o...
1390 13 D0 6F C9 E1 C9 79 E1 C1 E5 C5 4F CD AB 13 EB ..o...y....O....
13A0 E3 CD EB 14 D1 21 00 00 3E 01 C9 CD 58 10 2D 06 .....!..>...X.-.
13B0 21 00 00 C3 DD 13 CD 58 10 2B 00 CD E7 13 CD 58 !......X.+.....X
13C0 10 2B 15 E5 CD E7 13 EB E3 7C AA 7A 19 D1 FA BE .+.......|.z....
13D0 13 AC F2 BE 13 C3 96 10 CD 58 10 2D 92 E5 CD E7 .........X.-....
13E0 13 CD D6 14 C3 C7 13 CD 4B 14 CD 58 10 2A 2D E5 .........K..X.*-.
13F0 CD 4B 14 06 00 CD D3 14 E3 CD D3 14 EB E3 7C B7 .K.............|.
1400 CA 09 14 7A B2 EB C2 97 10 7D 21 00 00 B7 CA 3D ...z.....}!.....=
1410 14 19 DA 97 10 3D C2 11 14 C3 3D 14 CD 58 10 2F ......=....=..X./
1420 4E E5 CD 4B 14 06 00 CD D3 14 E3 CD D3 14 EB E3 N..K............
1430 EB 7A B3 CA 97 10 C5 CD B6 14 60 69 C1 D1 7C B7 .z........`i..|.
1440 FA 96 10 78 B7 FC D6 14 C3 EA 13 21 FE 17 C3 79 ...x.......!...y
1450 18 CD 24 10 DA 5C 14 7E 23 66 6F C9 CD 6C 10 78 ..$..\.~#fo..l.x
1460 B7 C0 CD 58 10 28 09 CD 63 13 CD 58 10 29 01 C9 ...X.(..c..X.)..
1470 C3 27 15 CD 62 14 7C B7 FA 96 10 B5 CA 96 10 D5 .'..b.|.........
1480 E5 2A 19 20 11 05 20 CD 0F 10 DA 90 14 21 C0 16 .*. .. ......!..
1490 5E 23 56 22 19 20 E1 EB C5 CD B6 14 C1 D1 23 C9 ^#V". .........#.
14A0 CD 62 14 1B CD D3 14 13 C9 2A 1B 20 D5 EB 2A 48 .b.......*. ..*H
14B0 21 CD CC 14 D1 C9 E5 6C 26 00 CD C1 14 41 7D E1 !......l&....A}.
14C0 67 0E FF 0C CD CC 14 D2 C3 14 19 C9 7D 93 6F 7C g...........}.o|
14D0 9A 67 C9 7C B7 F0 7C B5 C8 7C F5 2F 67 7D 2F 6F .g.|..|..|./g}/o
14E0 23 F1 AC F2 96 10 78 EE 80 47 C9 7C AA F2 F1 14 #.....x..G.|....
14F0 EB CD 0F 10 C9 CD 24 10 DA 27 15 E5 CD 58 10 3D ......$..'...X.=
1500 0A CD 63 13 44 4D E1 71 23 70 C9 C3 27 15 CD 58 ..c.DM.q#p..'..X
1510 10 3B 04 F1 C3 59 11 CD 58 10 0D 04 F1 C3 49 11 .;...Y..X.....I.
1520 C9 CD 15 10 FE 0D C8 D5 11 AA 10 97 CD CA 15 D1 ................
1530 1A F5 97 12 2A 07 20 E5 7E 23 B6 D1 CA 88 10 7E ....*. .~#.....~
1540 B7 FA EB 12 CD 4C 16 1B F1 12 3E 3F CD FB 16 97 .....L....>?....
1550 CD CA 15 C3 B8 10 D5 11 B1 10 C3 2B 15 CD FB 16 ...........+....
1560 CD A9 18 CD 20 17 FE 01 CA 88 15 CD FB 16 FE 0A .... ...........
1570 CA 63 15 B7 CA 63 15 FE 1B CA 98 15 12 13 FE 0D .c...c..........
```


Abb. 3.3-2

```
1580 C8 7B CD B9 18 C2 63 15 7B CD C0 18 CA 98 15 1B .{....c.{.......
1590 3E 08 CD FB 16 C3 63 15 CD F9 16 3E 0B C3 5D 15 >.....c....>..].
15A0 7C B7 FA 96 10 11 4E 21 E5 2A 1B 20 2B CD 0F 10 ¦.....N!.*. +...
15B0 E1 D8 1A 95 47 13 1A 9C DA BF 15 1B B0 C9 13 13 ....G...........
15C0 1A FE 0D C2 BF 15 13 C3 A8 15 47 1A 13 B8 C8 CD ...........G....
15D0 FB 16 FE 0D C2 CB 15 C9 CD 58 10 22 0F 3E 22 CD ..........X.".>".
15E0 CA 15 FE 0D E1 CA 49 11 23 23 23 E9 CD 58 10 27 ......I.###..X.'
15F0 05 3E 27 C3 DF 15 CD 58 10 5F 0C 3E 8D CD FB 16 .>'....X._.>....
1600 CD FB 16 E1 C3 E8 15 C9 06 00 CD D3 14 F2 13 16 ................
1610 06 2D 0D D5 11 0A 00 D5 0D C5 CD B6 14 78 B1 CA .-............x..
1620 2A 16 E3 2D E5 60 69 C3 1A 16 C1 0D 79 B7 FA 39 *..-.`i.....y..9
1630 16 3E 20 CD FB 16 C3 2B 16 78 B7 C4 FB 16 5D 7B .> ....+.x....]{
1640 FE 0A D1 C8 C6 30 CD FB 16 C3 3F 16 1A 6F 13 1A .....0....?..o..
1650 67 13 0E 04 CD 08 16 3E 20 CD FB 16 97 CD CA 15 g......> ........
1660 C9 CD 0F 10 C8 1A 02 13 03 C3 61 16 78 92 C2 74 ...........a.x..t
1670 16 79 93 C8 1B 2B 1A 77 C3 6C 16 C1 E1 22 0F 20 .y...+.w.l....".
1680 7C B5 CA 95 16 E1 22 11 20 E1 22 13 20 E1 22 15 ¦......". .". .".
1690 20 E1 22 17 20 C5 C9 21 45 20 CD D6 14 C1 39 D2  .". ..!E ....9.
16A0 56 15 2A 0F 20 7C B5 CA BD 16 2A 17 20 E5 2A 15 V.*. ¦....*. .*.
16B0 20 E5 2A 13 20 E5 2A 11 20 E5 2A 0F 20 E5 C5 C9  .*. .*. .*. ...
16C0 31 0F 21 3E FF 32 06 20 16 03 CD F9 16 15 C2 CA 1.!>.2. ........
16D0 16 97 11 3A 17 CD CA 15 21 C0 16 22 19 20 21 4E ...:....!..". !N
16E0 21 22 1B 20 21 A8 23 22 48 21 21 AA 23 22 4A 21 !". !.#"H!!.#"J!
16F0 21 EA 23 22 4C 21 C3 B8 10 3E 0D C5 F5 3A 06 20 !.#"L!...>...:.
1700 B7 C2 07 17 F1 C1 C9 F1 F5 4F 79 FE 0D CA 16 17 ..........Oy.....
1710 CD 09 10 F1 C1 C9 0E 0D CD 09 10 0E 0A C3 10 17 ................
1720 CD 06 10 E6 7F FE 02 C2 34 17 3A 06 20 2F 32 06 .........4.:. /2.
1730 20 C3 20 17 FE 03 C0 C3 B8 10 52 44 4B 20 50 52 . ........RDK PR
1740 4F 4D 50 54 20 42 41 53 49 43 20 56 33 2E 32 20 OMPT BASIC V3.2
1750 33 4B 0D 0A 4C 49 53 54 00 73 11 52 55 4E 00 43 3K..LIST.s.RUN.C
1760 11 4E 45 57 00 34 11 42 59 45 00 F8 18 45 4E 44 .NEW.4.BYE...END
1770 00 C7 18 4E 45 58 54 00 73 12 4C 45 54 00 55 13 ...NEXT.s.LET.U.
1780 49 46 00 DA 12 47 4F 54 4F 00 62 11 47 4F 53 55 IF...GOTO.b.GOSU
1790 42 00 D1 11 52 45 54 55 52 4E 00 F3 11 52 45 4D B...RETURN...REM
17A0 00 D4 12 46 4F 52 00 0E 12 49 4E 50 55 54 00 F5 ...FOR...INPUT..
17B0 12 50 52 49 4E 54 00 8B 11 53 54 4F 50 00 3D 11 .PRINT...STOP.=.
17C0 43 41 4C 4C 00 FC 18 4F 55 54 43 48 41 52 00 01 CALL...OUTCHAR..
17D0 1A 4F 55 54 00 09 19 4F 24 00 5A 19 49 24 00 67 .OUT...O$.Z.I$.g
17E0 19 50 4F 4B 45 00 A0 19 54 41 42 00 2F 19 42 59 .POKE...TAB./.BY
17F0 54 45 00 B5 19 57 4F 52 44 00 BF 19 00 4F 13 52 TE...WORD....O.R
1800 4E 44 00 73 14 41 42 53 00 A0 14 53 49 5A 45 00 ND.s.ABS...SIZE.
1810 A9 14 50 45 45 4B 00 99 19 49 4E 43 48 41 52 00 ..PEEK...INCHAR.
1820 0B 1A 48 45 58 00 12 1A 49 4E 00 40 19 27 00 E9 ..HEX...IN.@.'..
1830 19 54 4F 50 00 F7 19 4C 45 4E 00 FC 19 43 53 54 .TOP...LEN...CST
1840 53 00 E1 19 00 51 14 54 4F 00 1E 12 00 27 15 53 S....Q.TO....'.S
1850 54 45 50 00 2A 12 00 30 12 3E 3D 00 6D 13 23 00 TEP.*..0.>=.m.#.
1860 73 13 3E 00 79 13 3D 00 88 13 3C 3D 00 80 13 3C s.>.y.=...<=...<
1870 00 8E 13 00 94 13 21 53 17 CD 15 10 D5 1A 13 FE .......!S........
1880 2E CA 9B 18 23 BE CA 7D 18 3E 00 1B BE CA A2 18 ....#..}.>......
1890 23 BE C2 90 18 23 23 D1 C3 79 18 3E 00 23 BE C2 #....##..y.>.#..
18A0 9D 18 23 7E 23 66 6F F1 E9 E5 2A 4A 21 54 5D E1 ..#~#fo...*J!T].
18B0 C9 E5 2A 48 21 54 5D E1 C9 E5 2A 4C 21 BD E1 C9 ..*H!T]...*L!...
18C0 E5 2A 4A 21 BD E1 C9 CD 63 13 EB 21 A8 23 EB CD .*J!....c..!.#..
18D0 0F 10 DA 57 15 7C B7 FA 57 15 7E 2F 77 46 B8 C2 ...W.¦..W.~/wF..
18E0 57 15 22 4C 21 7D D6 84 6F 7C DE 00 67 22 4A 21 W."L!}..o¦..g"J!
18F0 2B 2B 22 48 21 C3 B8 10 FF C3 B8 10 CD 63 13 D5 ++"H!.........c..
1900 01 05 19 C5 E9 D1 CD 1D 10 CD 62 14 E5 CD 58 10 ............b...X.
1910 3D 1A CD 63 13 45 3E D3 32 02 20 E1 7D 32 03 20 =..c.E>.2. .}2.
1920 3E C9 32 04 20 78 CD 02 20 CD 1D 10 C3 27 15 CD >.2. x.. ....'..
1930 62 14 7C B5 CC 1D 10 2B 3E 20 CD FB 16 C3 32 19 b.¦....+> ....2.
1940 CD 62 14 E5 3E DB 32 02 20 E1 7D 32 03 20 3E C9 .b..>.2. .}2. >.
1950 32 04 20 CD 02 20 26 00 6F C9 CD 63 13 D5 EB AF 2. .. &.o..c....
```

Abb. 3.3-2

```
1960 CD CA 15 D1 CD 1D 10 CD 63 13 D5 EB 2A 1B 20 EB .........c...*. .
1970 CD 0F 10 DA 57 15 CD A9 18 CD 63 15 44 4D EB 2B ....W.....c.DM.+
1980 CD A9 18 D5 CD 61 16 AF 02 D1 23 CD CC 14 EB 21 ......a....#....!
1990 00 20 73 23 72 D1 CD 1D 10 CD 62 14 6E 26 00 C9 . s#r......b.n&..
19A0 CD 63 13 E5 CD 58 10 2C 09 CD 63 13 7D E1 77 CD .c...X.,..c.}.w.
19B0 1D 10 C3 27 15 CD 62 14 7D CD CD 19 CD 1D 10 CD ...'..b.}.......
19C0 62 14 7C CD CD 19 7D CD CD 19 CD 1D 10 F5 0F 0F b.|...}.........
19D0 0F 0F CD D6 19 F1 E6 0F C6 90 27 CE 40 27 C3 FB ...........'.@'..
19E0 16 CD 0C 10 2F 6F 26 00 C9 1A 13 6F 26 00 CD 58 ..../o&....o&..X
19F0 10 27 01 C9 C3 27 15 2A 1B 20 23 C9 2A 00 20 2B .'...'.*. #.*. +
1A00 C9 CD 63 13 7D CD FB 16 CD 1D 10 CD 20 17 26 00 ..c.}....... .&.
1A10 6F C9 C5 21 00 00 CD 58 10 28 1D 1A FE 0D CA 27 o..!...X.(......'
1A20 15 CD 3D 1A 29 29 29 29 06 00 4F 09 13 CD 58 10 ..=.))))..O...X.
1A30 29 03 C3 3B 1A C3 1B 1A C3 27 15 C1 C9 FE 30 FA )..;......'....0.
1A40 27 15 FE 39 FA 54 1A CA 54 1A FE 41 FA 27 15 FE '..9.T..T..A.'..
1A50 47 F2 27 15 D6 30 FE 0A F8 D6 07 C9 CD 0C 10 C8 G.'..0..........
1A60 CD 06 10 FE 03 C0 C3 B8 10 00 00 00 00 00 00 00 ................
1A70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

Abb. 3.3-2

3.3.1 I/O-Routinen

Der Basic-Interpreter muß irgendwie mit seiner Umwelt Kontakt aufnehmen können. Er soll Meldungen von einer Konsole empfangen können und Antworten auf dieser Konsole wieder ausgeben können. Die Konsole kann dabei über eine serielle Datenleitung oder über eine parallele Verbindung mit dem Computer verbunden sein. Der Basic-Interpreter benötigt eine Schnittstelle, bei der ein Zeichen zur Konsole gesendet wird und eine, bei der ein Zeichen empfangen wird.

Damit jeder Benutzer seine Zeichenein- und Ausgaberoutine selbst schreiben und sehr leicht in das Basic einbauen kann, geschieht der Anschluß an diese Routinen über Sprungvektoren. Auf Adresse 1006H bis 100CH (H steht für Hexa- oder Sedezimal) steht eine solche Sprungtabelle.

Der erste Sprung führt auf ein Unterprogramm mit der Aufgabe, ein Zeichen von der Konsole zu holen. Dabei muß dieses Unterprogramm solange warten, bis ein Zeichen eingegeben wird und dann das Zeichen im Register A abliefern. Andere Register dürfen nicht verändert werden. Der Code des Zeichens muß ASCII sein (ISO-7-Bit-Code nach DIN 66003).

Die Routine ECHO hier auf Adresse 1009H springt auf ein Unterprogramm, das ein Zeichen auf der Konsole ausgeben muß. Das Zeichen befindet sich im Register C und sollte nach dem Rücksprung (RET) auch im Register A stehen. Andere Register dürfen nicht verändert werden.

Nun folgt eine Routine mit der Marke CSTS. Der Sprung erfolgt auf Adresse 100CH. Das Unterprogramm hat die Aufgabe zu melden, ob ein Zeichen von der Konsole eingegeben wurde. Dies ist nötig, um zum Beispiel während eines Programmlaufs festzustellen, ob CTRL-C betätigt wurde und der Abbruch des Basic-Programmlaufs erfolgen soll, weil z.B. das Programm in eine Schleife geraten ist. Das Unterprogramm liefert im Register A den Wert 0 , falls kein Zeichen eingegeben wurde und 0FFH, falls ein Zeichen eingegeben wurde. Die Flags müssen dabei gesetzt sein (spez. das Zero-Flag).

Die Unterprogrammsprünge wurden hier für den ZAPPLE-Monitor angegeben [5]. Das Unterprogramm CSTS wird in der Routine CONT gebraucht. Diese Routine auf der Adresse 1A5CH im Listing hat die Aufgabe, festzustellen, ob CTRL-C betätigt wurde. CTRL-C hat den Code 03. Zu-

nächst wird durch den Aufruf von CSTS festgestellt, ob überhaupt ein Zeichen eingegeben wurde; falls nicht, erfolgt durch den Befehl RZ ein Rücksprung. Sonst wird durch den Aufruf CI ein Zeichen von der Konsole geholt und verglichen, ob es sich um CTRL-C gehandelt hat. Falls nicht, so erfolgt ebenfalls ein Rücksprung; sonst wird auf RSTART gesprungen, um das Basic neu zu starten. Dabei ist RSTART nicht mit START zu verwechseln: Wird das Basic auf der Marke START gestartet, so werden bisherige Programme gelöscht. Bei RSTART werden die alten Werte übernommen, und ein zuvor eingegebenes Basic-Programm bleibt erhalten.

Ein weiteres I/O-Programm ist OUTC (Adresse 16FBH). Es hat im Prinzip die gleiche Aufgabe wie ECHO, nämlich ein Zeichen auf der Konsole auszugeben. Dabei ist die Aufgabenstellung aber zusätzlich verfeinert: Ein Zeichen wird nur dann ausgegeben, wenn OCSW, eine Variable, einen Wert ungleich 0 hat; außerdem wird, falls der Wert 0DH (Wagenrücklauf) ausgegeben werden soll, zusätzlich auch 0AH (Zeilenvorschub) ausgegeben.

Die Variable OCSW ist nötig, um später beim Laden von Programmen, z.B. durch eine Kassette, den Ausdruck zu unterbinden, um dadurch Zeit zu sparen.

CHKIO auf Adresse 1720H dient der Eingabe von Zeichen. Dabei wird auf die Steuerzeichen CTRL-C (Code 03) und CTRL-B (Code 02) geachtet. CTRL-B schaltet OCSW jeweils in den komplementären Zustand, so daß, falls vorher die Ausgabe freigegeben war, sie jetzt unterbunden wird und umgekehrt.

Wurde CTRL-C betätigt, so wird das Basic durch einen Sprung auf die Adresse RSTART erneut gestartet. Ebenfalls wird durch die Routine das Paritätsbit mit dem Befehl „ANI 7FH" wegmaskiert, so daß nur der reine ASCII-Code übrigbleibt. Dies ist nötig, um Zeichen überhaupt erkennen zu können, insbesondere, wenn unterschiedliche Konsolen verwendet werden, die meist ein Parity-Bit mitsenden, dabei entweder die gerade (even), die ungerade (odd) Parität oder überhaupt keine Parität.

3.3.2 Einfache zeichenverarbeitende Routinen

Als Konvention für den Interpreter wurde festgelegt, daß das Registerpaar DE als Zeiger auf den Basic-Text wirkt. Das Unterprogramm IGNB auf Adresse 1015H hat die Aufgabe, Leerzeichen, die in Basic-Programmen vorkommen, zu ignorieren. Dazu wird in der Routine die von DE adressierte Speicherzelle mit dem ASCII-Wert für das Leerzeichen verglichen (20H).

Falls der Wert nicht gleich hex 20 ist, wird ein Rücksprung ausgeführt; andernfalls wird der Inhalt des Registerpaars DE um eins erhöht, um auf die nächste Speicherzelle zu zeigen. Es folgt dann ein Sprung auf die Routine selbst; somit kehrt das Unterprogramm erst dann zurück, wenn kein Leerzeichen mehr gefunden wird.

TSTC auf Adresse 1058H stellt eine der am häufigsten gebrauchten Routine dar. Sie wird verwendet, um ein Zeichen im Basic-Textbuffer mit einem bestimmten Wert zu vergleichen, und außerdem, um Leerzeichen zu ignorieren. Das Zeichen, mit dem verglichen werden soll, steht direkt hinter dem Unterprogrammaufruf. Als weiterer Parameter wird eine relative Sprungadresse mit angegeben, auf die gesprungen wird, falls das Zeichen nicht im Basic-Buffer stand. TSTV auf Adresse 1024H stellt fest, ob sich im Buffer eine numerische Variable befindet.

Eine derartige Variable kann ein Buchstabe zwischen A und Z oder das Zeichen „@" sein, das für ein eindimensionales Feld steht. Es erhält als Parameter den Index. Als Ergebnis steht im Registerpaar HL die Adresse der Variablen und es wird, falls es keine Variable war, das Carry-Flag gesetzt. Dazu wird zunächst mit dem Pro-

grammteil IGNB nach dem ersten Zeichen gesucht, das kein Leerzeichen ist. Durch die Subtraktion des Wertes 40H wird festgestellt, ob der Wert dieses Zeichens größer oder gleich 40H ist. Damit sind alle Sonder- und Steuerzeichen erkannt. Falls es nicht das Zeichen 40H war, wird auf die Marke TV1 gesprungen. Falls es das Zeichen 40H war, wird ein Programm PARN aufgerufen, dessen genaue Funktion später noch erklärt wird. Es liefert im Registerpaar HL den Wert, der dem in Klammern stehenden Wert eines arithmetischen Ausdrucks entspricht. Ein solcher Ausdruck kann z.B. wie folgt aussehen:

(A+3xB) oder (4) usw.

Dabei wird, falls in dieser Klammer wie oben die Variable A steht, natürlich wieder die Routine TSTV aufgerufen. Sie wird also rekursiv verwendet.

Dieser Wert soll als Index verwendet werden. Dazu muß nun die effektive Adresse der indizierten Variablen ausgerechnet werden. Dazu wird nach einigen Ende-Prüfungen durch Subtraktion des zweifachen Wertes von der Adresse des Textendes die Adresse berechnet. Die Variablen des Feldes laufen also von oben nach unten. Die Multiplikation mit 2 durch den Befehl DAD H ist nötig, weil immer zwei Bytes für eine Variable benötigt werden. Der Unterprogrammaufruf SIZE bewirkt die Feststellung des belegten Speicherbereichs durch das Basic-Programm.

Bei TV1 ist die Situation einfacher. Der gewonnene Wert der Variable, der zwischen 1 und 26 liegt, wird mit 2 multipliziert und auf die Adresse VARBGN addiert. Daraus ergibt sich direkt die Speicheradresse.

TSTNUM auf Adresse 106CH muß den Wert eines numerischen Wertes im Register HL abliefern. Dazu ist eine Dezimal-nach-Binär-Umwandlung nötig. Die Zahl wird im Zweierkomplement mit 16 bit dargestellt. Das Vorzeichen der Zahl wird in anderen Programmteilen festgestellt. Die Umwandlung in den Binärcode erfolgt durch Multiplikation mit dem Wert 10 und Addition der nachfolgenden Zahl:

HL:=HL*10+neue Zahl

Die Multiplikation mit 10 wird durch eine Aufteilung der Zahl erreicht:

10:=((2*2)+1)*2

somit ergibt sich:

HL:= ((HL*2*2)+HL)*2

Eine Multiplikation mit 2 ist aber einfach durch den Befehl DAD H erreichbar. Die Addition von HL wird mit einem DAD B Befehl erreicht, wobei der ursprüngliche HL-Wert, der dazu benötigt wird, zuvor in das Register BC gebracht wurde.

Die Sequenz:
```
MOV B,H
MOV C,L
DAD H
DAD H
DAD B
DAD H
```

bewirkt, daß der Inhalt des Registerpaars HL mit 10 multipliziert wird.

PRTNUM auf Adresse 1608H hat genau die umgekehrte Aufgabe: Es muß einen im Registerpaar stehenden Wert ausgeben. Dabei steht im Register C die Anzahl der Gesamtstellen. Dies ist für den Formatierbefehl wichtig. *Abb. 3.3.2-1* zeigt das dazugehörige Flußdiagramm. Als erstes wird das Vorzeichen der Zahl überprüft. Ist es negativ, so wird der Inhalt des Registerpaars HL komplementiert und der ASCII-Wert des Zeichens „–" in das Register B gespeichert, andernfalls der Wert 0. Ferner wird, falls der Wert von HL negativ ist, der Zeichenzähler C um eins verringert.

Das Registerpaar DE wird auf den Stack gerettet. Ebenfalls BC, das das Vorzeichen enthält. DE wird zuvor noch mit dem Wert 10 belegt und auf den Stack gelegt. Es erfolgt nun eine Division durch 10, also dem umgekehrten Algorithmus von vorher.

Mit der Sequenz „XTHL ...“ wird jeweils der Vorzeichenwert und der Zähler zurückgebracht und dann wieder auf den Stack gelegt in umgekehrter Folge. Dabei wird solange dividiert, bis der Rest zu Null geworden ist. Dann wird mit einer umgekehrten Sequenz der Stack wieder entleert und zuerst das Vorzeichen nach einer Folge von Blanks ausgegeben, dann folgen die Werte. Der zuvor abgelegte Wert 10, der in den Daten nicht vorkommen kann, wird als Endezeichen verwendet; danach erfolgt der Rücksprung. Das Arbeiten über den Stack ist nötig, da bei diesem Divisionsalgorithmus die Dezimalwerte von der niederen zur höheren Stelle erscheinen, aber umgekehrt ausgegeben werden müssen.

GETLN auf Adresse 155DH holt eine Zeile vom Benutzer und speichert diese, beginnend bei der indirekten Adresse, die durch BUFFER gegeben ist, ab.

Dazu wird zunächst das in A mitgegebene Zeichen durch den Aufruf CALL OUTC ausgegeben. Damit kann zum Beispiel das Meldezeichen „>“ oder „? “ bei einem INPUT mit ausgegeben werden. Danach wird ein Unterprogramm mit dem Namen DBUFF aufgerufen. Es hat die Aufgabe, in das Registerpaar DE die effektive Adresse des Textbuffers zu holen. Die Adresse steht dabei in zwei aufeinanderfolgenden Bytes auf der Adresse BUFFER.

Hinter der Marke GL1 wird dann ein Zeichen vom Benutzer geholt. Handelte es sich um das Zeichen mit dem ASCII-Code 1, so wird nach GL3 weitergesprungen. Damit kann ein zuvor falsch eingegebenes Zeichen gelöscht werden. Der Pointer DE wird dazu um Eins dekrementiert, falls die Überprüfung durch CXBUFA nicht ergab, daß ein weiteres Löschen nicht mehr möglich war, da es das erste Zeichen im Buffer ist.

Bei allen anderen Zeichen wird der Code ausgegeben. Danach erfolgt ein Vergleich auf LF (Code ϕAH). Dieses Zeichen wird nicht im Buffer abgelegt, ebenfalls nicht

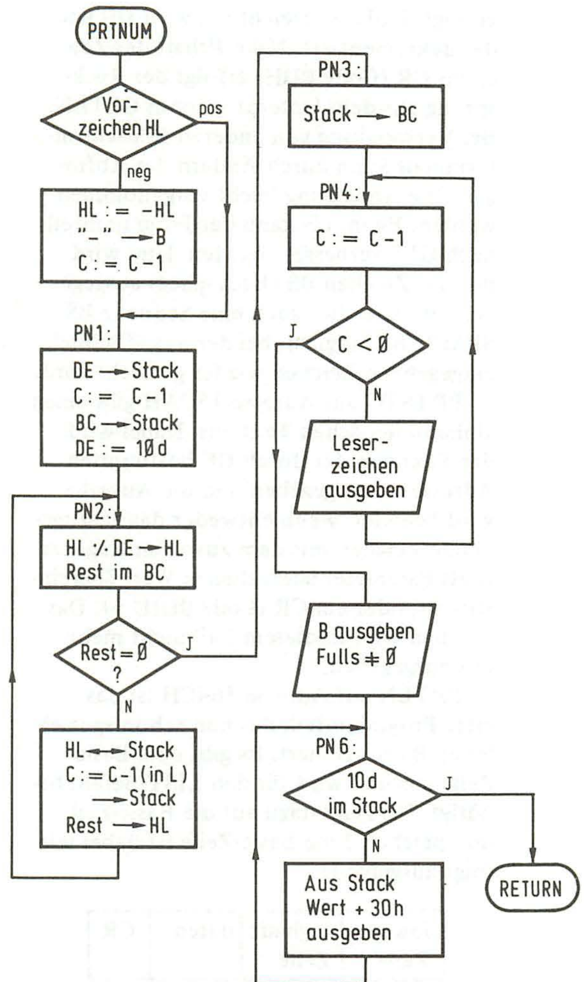


Abb. 3.3.2-1 Flußdiagramm für PRTNUM

ein Zeichen mit dem Code ϕϕ (NUL). Mit ESC (Code 1BH) kann die ganze Zeile gelöscht werden. Es wird CRLF ausgegeben und das Zeichen ϕBH, das den Cursor um eine Zeile nach oben positionieren soll.

In jedem anderen Fall wird das Zeichen auf der Adresse DE abgespeichert und DE um eins inkrementiert. Durch den Aufruf CXBUFE wird auf das Ende des Buffers ab-

gefragt. Falls es erreicht ist, wird DE wieder dekrementiert. Nach Erhalt des Zeichens CR (Code ØDH) erfolgt der Rücksprung aus dem Unterprogramm GETLN. Bei Verwendung von anderen Bildschirmterminals kann durch Ändern der Abfragen eine Anpassung leicht vorgenommen werden. Ebenfalls kann der Programmteil nach GL3 verbessert werden. Hier wird nur das Zeichen BS (Backspace) ausgegeben; es wäre aber auch eine Sequenz BS SPACE BS möglich, bei der das fälschlich eingegebene Zeichen wieder gelöscht wird.

PRTSTG auf Adresse 15CAH gibt einen alphanumerischen Text aus. Dabei wird der Text von der durch DE bestimmten Adresse an ausgegeben und die Ausgabe wird beendet, wenn entweder das auszugebende Zeichen mit dem zuvor im Register A als Parameter übergebenen Wert übereinstimmt, oder ein CR (Code ØDH) ist. Das Zeichen wird in diesem Fall nicht mehr mit ausgegeben.

PRTLN auf Adresse 164CH ist das erste Programmteil, das nun schon spezieller an Basic erinnert. Es gibt eine Basic-Zeile aus und wird für den LIST-Befehl benötigt. DE zeigt dazu auf die Basic-Zeile im Speicher. Eine Basic-Zeile ist dabei wie folgt aufgebaut:

lowadr Zeile	highadr Zeile	daten	CR

Im Binärcode steht am Anfang die Zeilennummer; sie belegt zwei Bytes. Die Nummer kann von 1 bis 32767 reichen. Größere Zahlen sind nicht möglich, da sie negativ ausgegeben würden und nicht konsistent mit den übrigen Interpreterteilen wären. Das Ende der Zeile wird durch ein CR angekündigt.

In PRTLN wird zunächst die Zeilennummer in das Registerpaar HL geholt und mit PRTNUM ausgegeben. DE zeigt dabei immer auf die Stelle der Basic-Zeile, die gerade angesprochen wird. Es wird dann ein Blank (Leerzeichen) ausgegeben, dann (mit dem zuvor besprochenen Unterprogramm PRTSTG) der Rest der Zeile. Das Register A vor dem Eintritt in das Unterprogramm mit dem Wert 0 belegt durch die Operation „SUB A", um zu verhindern, daß die Ausgabe vorzeitig unterbrochen wird, da PRTSTG jedes auszugebende Zeichen mit dem Registerinhalt von A vergleicht und die Ausgabe beendet, falls eine Übereinstimmung besteht. Der Code 0 kommt aber in Basic-Programmen nicht vor, was auch in der Routine GETLN explizit verhindert wurde, und kann somit als „dummy"-Wert verwendet werden.

3.3.3 Analyse von arithmetischen Ausdrücken

Eine der wichtigsten Fähigkeiten von einer höheren Programmiersprache, ist die Möglichkeit, Formeln auswerten zu können. Da dieser Abschnitt im Prinzip unabhängig von der Sprache Basic ist, soll er zuerst behandelt werden.

Beispiele für arithmetische Ausdrücke:

```
2+3
4*5+(2+3*ABS(-5))*2
```

Dabei können in den arithmetischen Ausdrücken auch Funktionen, wie z.B. ABS (Absolutbetrag) vorkommen. Unser Analysator kann nur ganzzahlige Arithmetik betreiben. Es wird dabei ein Zahlenbereich von +32767 und −32768 überdeckt. Der Bereich läßt sich mit 16 bit im Zweierkomplement darstellen. Daher sind natürlich Funktionen wie SIN oder LOG nicht in dem Analysator zu finden. Der Vorteil der Ganzzahl-Arithmetik liegt in der hohen Ausführungsgeschwindigkeit.

Bei vielen Programmiersprachen ist die Möglichkeit vorhanden, den Zahlenbereich der verwendeten Arithmetik einzustellen, dies geschieht zum Beispiel mit den Befehlen INTEGER, REAL, DOUBLE PRECISION etc. Unsere Arithmetik entspricht dabei der Form INTEGER.

In arithmetischen Ausdrücken dürfen selbstverständlich auch Variablen verwendet werden, so daß ein Ausdruck auch wie folgt lauten kann:

2+A*B

A und B sind die dabei verwendeten Variablen. Als Variable kann auch das eindimensionale Feld „@" verwendet werden, das schon bei der Besprechung des Unterprogramms TSTV erklärt wurde.

Zur Erklärung des Programmablaufs werden von nun an auch sogenannte Syntaxdiagramme verwendet. Sie wurden erstmals bei der Sprachdefinition von Pascal eingesetzt und haben sich als sehr anschaulich erwiesen.

Abb. 3.3.3-1 zeigt die Elemente von Syntaxdiagrammen. Es werden dabei zwei verschiedene Symbole verwendet. Ein Name, der in einem Rechteck vorkommt, ist ein Beschreibungsaufruf, oder mit dem Fachwort ein „Nichtterminalsymbol". Ein Name in einem runden Symbol ist ein Endzeichen oder Terminalsymbol.

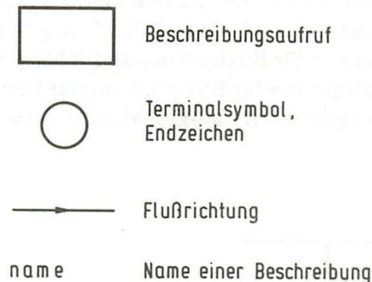


Abb. 3.3.3-1 Elemente von Syntaxdiagrammen

Dann gibt es noch einen Richtungsablauf, der die Leserichtung des Diagramms angibt, wie bei einem Flußdiagramm. Ein Name, der über einem Flußstrich steht, definiert eine Beschreibung oder ein Nichtterminalsymbol.

Mit diesen Syntaxdiagrammen läßt sich der prinzipielle Ablauf unseres Analysators genau definieren. Dabei sei noch bemerkt, daß beim Programmieren eine besondere Technik, die des rekursiven Aufrufs verwendet wird, eine Technik, die besonders zur Realisierung von Syntaxdiagrammen geeignet ist.

Rekursiv bedeutet, daß es zum Beispiel einem Unterprogramm erlaubt ist, sich selbst aufzurufen. Das Unterprogramm muß dann mit einem Abbruchkriterium dafür sorgen, daß dies nicht endlos geschieht, sondern nur, solange es eben zur Lösung einer Aufgabe erforderlich ist. Ein gutes Beispiel dafür ist die rekursive Definition der Fakultät. Für sie gilt FAC=1, falls das Argument kleiner oder gleich Null ist und FAC(I)=FAC(I-1)*I, falls nicht. Ein Beispiel: FAC(3) soll berechnet werden. Das Argument ist nicht Null, somit gilt FAC(3)=FAC(2)*3. Nun muß FAC(2) berechnet werden; es gilt erneut: FAC(2)=FAC(1)*2 somit gesamt FAC(3) =FAC(1)*2*3 FAC(1) ist aber FAC(0)*1 somit FAC(3)=FAC(0)*1*2*3 und für FAC(0) gilt nach dem Abbruchkriterium FAC(0)=1 also FAC(3)=1*1*2*3. Dieser Ausdruck ergibt ausgewertet 6, also FAC(3)=6.

Nun zurück zur Analyse von arithmetischen Ausdrücken. *Abb. 3.3.3-2* zeigt das Syntaxdiagramm für EXPR, dem Aufruf

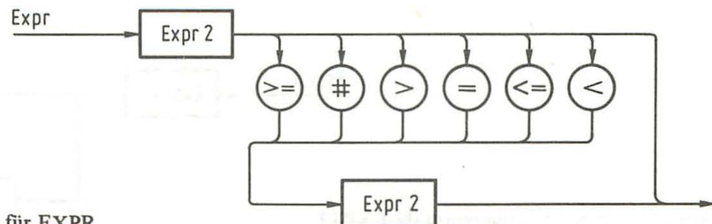


Abb. 3.3.3-2 Syntaxdiagramm für EXPR

des Analysators. EXPR kann sein EXPR2 gefolgt von „nichts", oder einem Vergleichsoperator der von EXPR2 gefolgt wird. Damit wird gesagt, daß ein arithmetischer Ausdruck aus einem arithmetischen Unterausdruck besteht, der ggf. von einem Vergleich mit einem weiteren Unterausdruck gefolgt werden kann.
Es ist also expr 2
oder expr2>expr2
zulässig. Als Vergleichsoperatoren sind GRÖSSER-GLEICH, UNGLEICH, GRÖSSER, GLEICH, KLEINER-GLEICH und KLEINER zulässig. Über die Auswertung wird in dem Syntaxdiagramm nichts weiter ausgesagt.

EXPR2 wird in *Abb. 3.3.3-3* definiert. EXPR2 kann sein wahlweise „+" oder „–" oder „nichts" gefolgt von EXPR3. Diese gefolgt von wahlweise „nichts", oder „+" oder „–" gefolgt von EXPR3, die wahlweise wieder gefolgt sein kann von „nichts", oder „+" oder „–" ..... . Damit kann folgender Aufbau erreicht werden:

+expr3
expr3
expr3+expr3–expr3
–expr3–expr3 usw.

Durch den Aufbau der Syntaxdiagramme wird hier auch die Vorrangfolge der mathematischen Zeichen festgelegt. Bisher gilt „+", „–" vor Vergleichsoperatoren, da die weiter innen liegenden Definitionsteile vor den äußeren Aufrufen ausgewertet werden. *Abb. 3.3.3-4* zeigt das Syntaxdiagramm für EXPR3. Hier wird die Verwendung von „*" und „/" erklärt. Sie ist fast analog zu „+" und „–".

EXPR4 zeigt *Abb. 3.3.3-5* EXPR4 kann sein eine Konstante, eine Variable, eine Funktion oder „KLAMMER AUF, EXPR, KLAMMER ZU." Der letzte Ausdruck verwendet wieder EXPR, womit die ganze Definition rekursiv geworden ist. Nun ist es möglich, alle Kombinationen von arithmetischen Ausdrücken zuzulassen, sie sind durch das Syntaxdiagramm genau festgelegt.

Abb. 3.3.3-6 definiert noch die Begriffe „Konstante" und „Ziffer". *Abb. 3.3.3-7* zeigt den Aufbau von Variablen. Dabei wird hier beim Sonderfall eines Feldes erneut EXPR rekursiv verwendet, womit auch klar ist, daß geschachtelte Aufrufe der Form FELD(FELD(FELD(...))) ebenfalls zulässig sind. *Abb. 3.3.3-8* zeigt das Diagramm für Buchstaben und *Abb. 3.3.3-9* das Diagramm für Funktionen, das hier nicht mehr weiter aufgeschlüsselt wird.

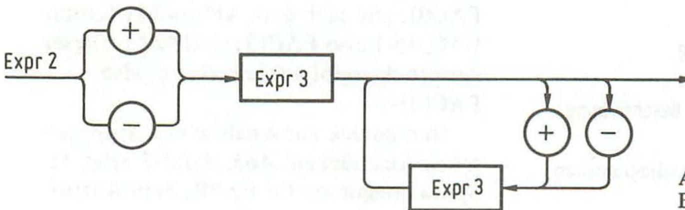


Abb. 3.3.3-3 Syntaxdiagramm für EXPR2

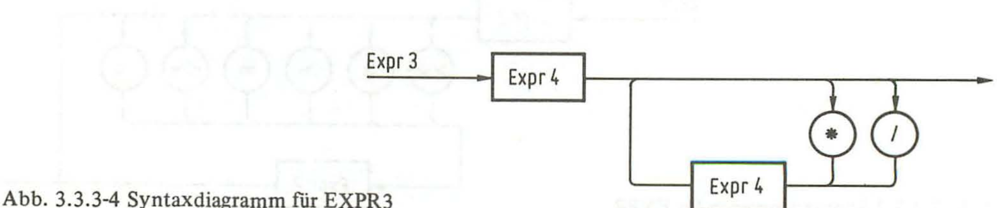


Abb. 3.3.3-4 Syntaxdiagramm für EXPR3

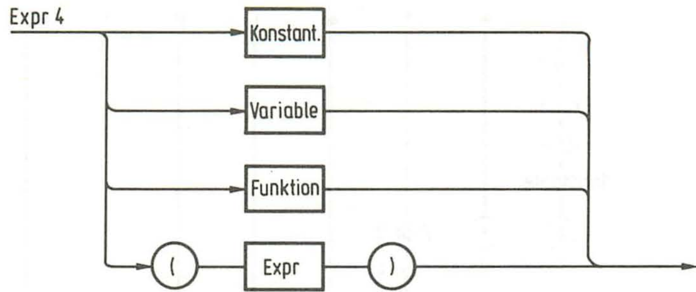
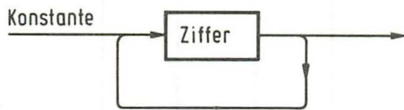


Abb. 3.3.3-5 Syntaxdiagramm für EXPR4



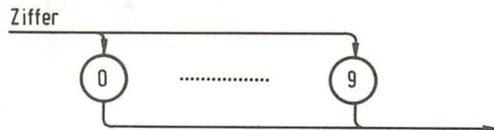


Abb. 3.3.3-6 Syntaxdiagramme für KONSTANTE und ZIFFER

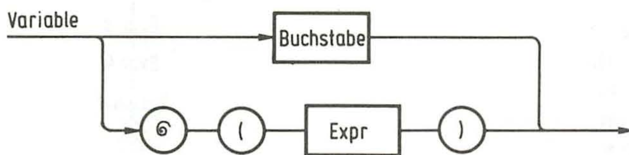


Abb. 3.3.3-7 Syntaxdiagramm für VARIABLE

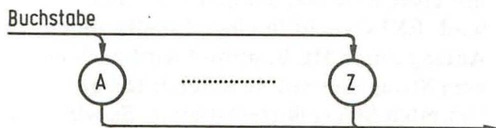


Abb. 3.3.3-8 Syntaxdiagramm für BUCHSTABE

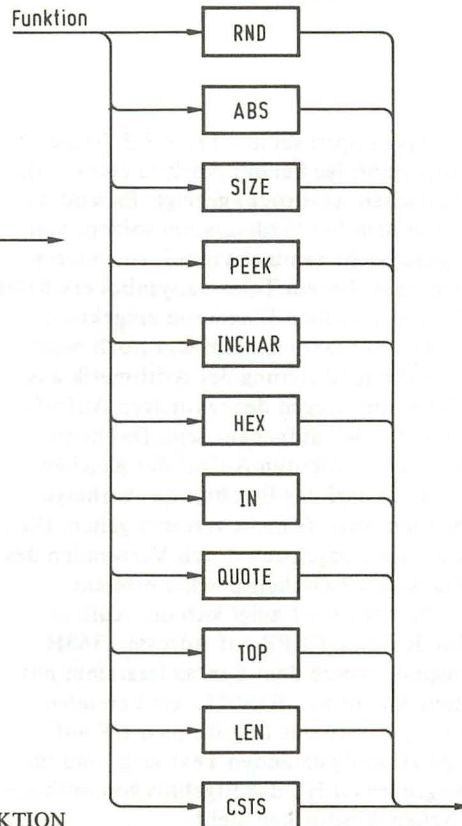


Abb. 3.3.3-9 Syntaxdiagramm für FUNKTION

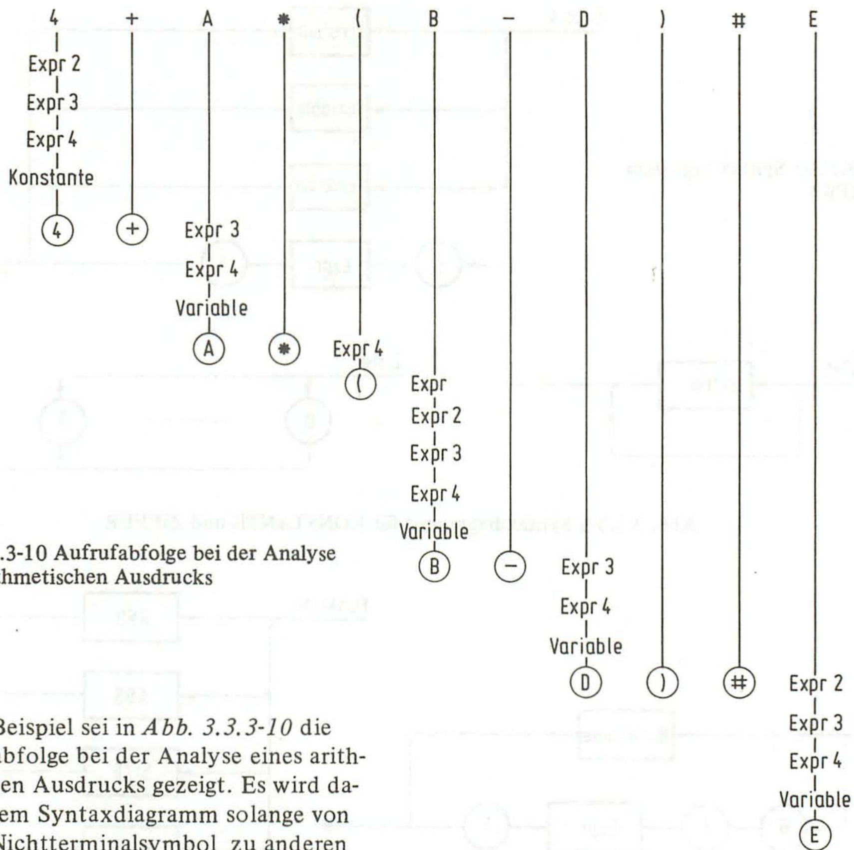


Abb. 3.3.3-10 Aufrufabfolge bei der Analyse eines arithmetischen Ausdrucks

Als Beispiel sei in *Abb. 3.3.3-10* die Aufrufabfolge bei der Analyse eines arithmetischen Ausdrucks gezeigt. Es wird dabei in dem Syntaxdiagramm solange von einem Nichtterminalsymbol zu anderen gegangen, bis ein Terminalsymbol erscheint. Dieses ist in dem Diagramm eingekreist.

Das Syntaxdiagramm sagt noch nichts über die Realisierung der Arithmetik aus. Diese muß wegen des rekursiven Aufrufs „reentrable" aufgebaut sein. Das heißt, bei einem erneuten Aufruf der gleichen Routine darf das Ergebnis des vorhergehenden Aufrufs nicht verloren gehen. Dies wird am einfachsten durch Verwenden des Stacks als Zwischenspeicher erreicht.

In Abb. 3.3-1 zeigt sich der Aufbau. Die Routine EXPR auf Adresse 1363H beginnt gemäß dem Syntaxdiagramm mit dem Aufruf von EXPR2. Als Vereinbarung gilt, daß das Registerpaar DE auf den zu analysierenden Text zeigt und im Registerpaar HL das Ergebnis von arithmetischen Ausdrücken steht.

Nach dem Aufruf von EXPR2 steht also in HL das Ergebnis. Nun muß nach Syntaxdiagramm untersucht werden, ob ein Vergleichsoperator folgt. Dies geschieht mit einer Routine, die EXEC genannt wird. EXEC sucht in einer Tabelle, deren Anfang durch HL bestimmt wird nach einem String, der mit dem durch DE bestimmten String übereinstimmt. Es wird dann auf eine hinter dem Suchwort stehende Adresse gesprungen. Falls kein Wort gefunden wird, so wird auf eine Ausweichadresse gesprungen.

In unserem Fall wird vor dem Aufruf von EXEC das Registerpaar HL gerettet, da dort das Ergebnis von EXPR2 steht und

dann HL mit der Adresse TAB8-1 geladen. Mit TAB8 beginnend stehen die Vergleichsoperatoren mit den dazugehörigen Ausführungsadressen. EXEC sucht, beginnend bei DE und TAB8, nach einem Vergleichsoperator. Wurde keiner gefunden, so wird die Routine XP17 angesprungen. Dort erfolgt mit dem Befehl POP H das Zurückholen des Ergebnisses und die Berechnung des arithmetischen Ausdrucks ist fertig. Die Routinen XP11 bis XP16 bedienen die einzelnen Fälle bei einem Vergleichsoperator. Als Beispiel „GRÖSSER-GLEICH":

Es wird XP11 angesprungen; dort erfolgt der Aufruf XP18, einem gemeinsamen Unterprogramm, in dem EXPR2 aufgerufen wird und mit einer allgemeinen Vergleichsroutine CKHLDE die Werte des ersten Aufrufs von EXPR2 und des jetzigen Aufrufs verglichen werden. Das Register A enthält den Wert 1, was bei unserer Arithmetik einem TRUE entspricht, wenn also der Vergleich richtig war. HL besitzt den Wert 0 bei Rückkehr aus XP18. In XP11 wird durch den Befehl „RC" erreicht, daß HL den Wert 0 besitzt, wenn „EXPR2a KLEINER EXPR2b" ist und sonst wird mit „MOV L,A" der Wert 1 in L geladen, falls EXPR2a GRÖSSER-GLEICH EXPR2b.

Ähnlich geschieht die Abarbeitung der anderen Vergleiche. Bei EXPR2 wird zunächst auf ein Vorzeichen abgeprüft. Bei einem negativen Vorzeichen wird EXPR3 über eine spezielle Routine XP26 aufgerufen, die erreicht, daß das Ergebnis von EXPR3 durch CHGSGN (change sign, Vorzeichenwechsel) negiert wird. Im Falle keines bzw. eines positiven Vorzeichens kann bei XP22, EXPR 3 direkt aufgerufen werden. Danach gibt es drei Fälle: Folgt ein „+" Symbol, dann wird über Xp24 die Addition ausgeführt; folgt ein „–" Symbol, dann wird subtrahiert über XP26. Bei keinem der beiden Symbole wird auf XP42 gesprungen und der RET-Befehl ausgeführt.

Ähnlich erfolgt die Behandlung von EXPR3; nur entfällt die Vorzeichenabfrage und Multiplikation und Divisionsaufrufe erfolgen. Die Ausführung von EXPR4 geschieht wieder mit Hilfe des Programmteils EXEC. Es wird diesmal TAB4 verwendet, die eine Liste aller verfügbaren Funktionsaufrufe enthält. Wurde eine dieser Funktionen gefunden, so wird zu dem dazugehörigen Programmteil gesprungen, sonst erfolgt die Ausführung von XP40. Dort wird als erstes untersucht, ob es eine Variable ist. Falls nicht, so wird mit XP41 auf einen numerischen Wert geprüft, war es dies ebenfalls nicht, so wird auf PARN verzweigt. Dort erfolgt die Prüfung von „(EXPR)". Fehlt die schließende Klammer, so wird QWHAT aufgerufen, um eine Fehlermeldung auszugeben. QHOW wird mehrfach im Fehlerfall aufgerufen, falls ein Überlauf stattfand.

Die einzelnen Funktionen

RND

Die Funktion RND wird auf Adresse 1473H behandelt. RND besitzt einen Parameter, der angibt, in welchem Bereich die Zufallszahl liegen soll. Die genaue Form lautet: RND(EXPR). Der Wert der Zufallszahl liegt zwischen 1 und EXPR. Dazu wird die Routine PARN aufgerufen, die genau der Forderung (EXPR) genügt. Im Registerpaar HL steht dann nach dem Aufruf der Wert von EXPR. Es wird zunächst das Vorzeichen geprüft: Wurde eine negative Zahl eingegeben, so wird QHOW angesprungen, ebenso, wenn der gesamte Ausdruck Null war. Als Zufallszahl wird der Inhalt des Programms selbst genommen; dazu steht der Pointer RANPNT zur Verfügung. Durch eine Division wird der Zufallswert in den richtigen Bereich gebracht und in HL übergeben.

ABS

ABS bildet den Absolutbetrag einer Zahl. Es wird mit dem Aufruf „CALL PARN"

ein Wert geholt. Dann wird das Vorzeichen geprüft und, falls die Zahl negativ ist, wird sie komplementiert. Dies geschieht automatisch in dem Unterprogramm CHKSGN.

SIZE
SIZE ist eine sogenannte Pseudovariable, da sie keinen Parameter besitzt und wie eine Variable verwendet werden kann, allerdings nur lesend. Beim Aufruf von SIZE wird der für Basic-Programme freie Speicherplatz berechnet und in HL übergeben.

PEEK
PEEK erhält als Parameter eine Adresse. Von dieser Adresse wird ein Byte geholt und als Wert übergeben. Dieser Befehl kann verwendet werden, um z.B. Daten aus einer Zelle zu holen, die durch ein Maschinenprogramm belegt wurde. Mit dem Aufruf PARN wird die Adresse geholt, die dadurch auch beliebig berechnet werden kann, der Inhalt davon nach Register L transportiert und H gelöscht.

INCHAR
INCHAR wirkt wieder wie eine Variable. Erfolgt der Aufruf, so wird genau ein Zeichen von der Konsole über die Routine CHKIO geholt. Dieses Zeichen wird aber nicht wieder ausgegeben, sondern gelangt als ASCII-Wert in das Register L, wobei H mit 0 belegt wird.
Damit ist zum Beispiel ein Ausdruck wie folgt möglich:

A = INCHAR+128

HEX
HEX erlaubt die Eingabe von sedezimalen Zahlen, um das maschinennahe Arbeiten zu erleichtern. Die genaue Form lautet:

HEX(Hexausdruck).

Hexausdruck ist dabei die sedezimale Zahl. Sie wird in dem Unterprogramm HEX durch CNVBN in einer Schleife HNXTH in eine Binärzahl umgewandelt. Als Ergebnis steht sie dann im Registerpaar HL.
Beispiel:

HEX(3FA).

IN
IN hat die Aufgabe, einen Wert von einem 8080-Port zu holen. Dazu wird mit dem Aufruf CALL PARN die Portadresse geholt. In einem externen RAM-Speicher wird nun ein selbst modifiziertes Programm vorbereitet, das die Sequenz „DB xx C9" erhält, wobei xx für den Kanal des Ports steht. Dies ist nötig, da dem 8080 keine Möglichkeit gegeben ist I/O-Ports indirekt anzusprechen, sondern nur mit festen Adressen, die im Adreßteil des Maschinenbefehls angegeben sind. Der Wert des Ports steht im Register L, H ist mit 0 besetzt.

QUOTE
Mit QUOTE kann ein ASCII-Zeichen in die dezimale Darstellung umgewandelt werden. Das Zeichen, das im Programm steht, wird dazu einfach in das Register L geladen und H mit 0 belegt. Es folgt dann noch eine Prüfung auf das abschließende Anführungszeichen. Ein Beispiel dafür:

'A' liefert den Wert 65
'B'−2 liefert den Wert 64.

Hier ist ausnahmsweise der Name des Programmteils nicht als Wort im Basic-Programm verwendet, sondern nur ein einfaches Anführungszeichen.

TOP
TOP ist wieder eine Pseudovariable. Das Ergebnis des Aufrufs ist der erste freie Platz hinter dem eingegebenen Basic-Programm. Diese Stelle wird durch TXTUNF verwaltet, worauf später noch genauer eingegangen wird.

LENGTH
LEN ist ebenfalls eine Pseudovariable. Sie liefert die Länge des Strings als Ergebnis, der zuletzt mit I$ eingegeben wurde. Die Länge steht in der Zelle LEGT. Der Aufruf von LENGTH erfolgt durch den Namen LEN.

CSTAT
Mit der Pseudovariablen CSTAT kann abgefragt werden, ob von der Konsole ein Zeichen eingegeben wurde; falls ja, so besitzt das Registerpaar HL den Wert 0.

3.3.4 Befehlsabarbeitung

Abb. 3.3.4-1 zeigt den groben Ablauf der Befehlsverarbeitung. Nach dem Aufruf von Start werden zunächst diverse Variable initialisiert. Es wird dabei der Basic-Programmspeicher gelöscht, indem der Programmzeiger zurückgesetzt wird. TXTUNF ist der Zeiger für das Textende des Basic-Programms. Er wird auf TXTBGN, den Textanfang, gesetzt. Die maximale Größe eines Programms wird durch den Inhalt der Variablen TXTEND bestimmt. Sie wird auf TXTE gelegt. Diese Variable kann vom Basic aus mit dem Befehl END verändert werden, um dynamisch größere Programme zuzulassen. BUFFER zeigt auf den Textzwischenspeicher, der bei der Eingabe einer Zeile verwendet wird. BUFEND gibt dementsprechend das Ende dieses Zwischenspeichers an.

Als nächstes folgt der Programmteil RSTART. Dorthin kann auch von einem Monitor aus gesprungen werden. Das vorhandene Programm bleibt erhalten. Es ist aber nicht möglich, auf dieses Programm zu springen, ohne daß zuvor mindestens einmal START angesprungen wurde, da sonst die Variablen undefiniert sind und dies zu einem nichtvorhersehbaren Verhalten des Interpreters führt.

Es wird, nachdem im Programmteil START die Überschrift und Kennung ausgegeben wurde, im Programmteil RSTART die Meldung „READY" ausgegeben.

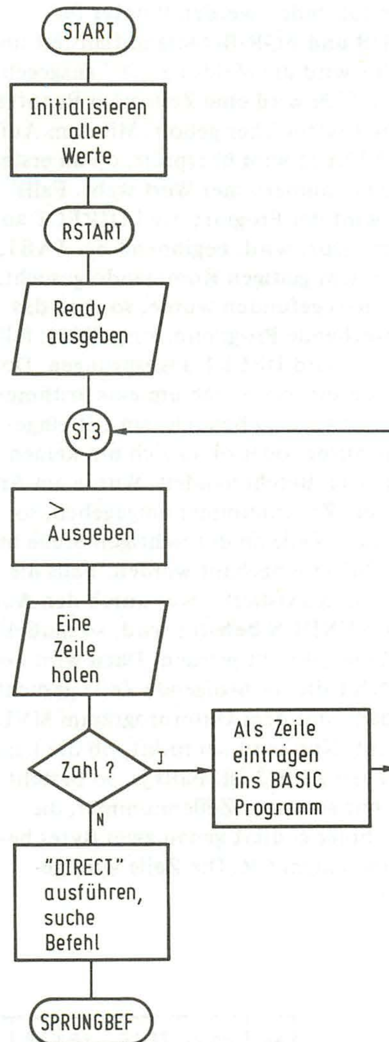


Abb. 3.3.4-1 Grober Ablauf der Befehlsverarbeitung

CURRNT bezeichnet die aktuelle Zeile, bei der die Programmausführung steht. Dieser Pointer wird auf den Wert „ST2+1" gesetzt und der Pointer zeigt dadurch auf 0.

Im folgenden werden Pointer für GOSUB und FOR-Befehle initialisiert und bei ST3 wird die Meldung „>" ausgegeben. Mit GETLN wird eine Zeile vom Benutzer in den Textspeicher geholt. Mit dem Aufruf TSTNUM wird überprüft, ob an erster Stelle ein numerischer Wert steht. Falls nein, wird der Programmteil DIRECT ausgeführt. Dort wird, beginnend bei TAB1, nach einem gültigen Kommando gesucht. Falls eines gefunden wurde, so wird das entsprechende Programm ausgeführt; falls nicht, so wird DEFLT angesprungen. Dort wird geprüft, ob es sich um eine arithmetische Zuweisung handelt, ein CR eingegeben wurde, oder ob es sich um keinen definierten Befehl handelt. Wurde am Anfang eine Zeilennummer eingegeben, so muß diese Zeile an der richtigen Stelle im Basic-Puffer eingebaut werden. Falls die Zeile schon existierte, was durch den Aufruf von FNDLN bewirkt wird, so muß die alte Zeile gelöscht werden. Dazu wird mit FNDNXT die nachfolgende Zeile gesucht und dann mit dem Unterprogramm MVUP gelöscht. Nun wird überprüft, ob die Länge der neuen Zeile 3 ist. Falls ja, so besteht diese nur aus einer Zeilennummer, die schon binär codiert genau zwei Bytes belegt und einem CR. Die Zeile wird gelöscht.

Falls die Zeile länger ist, so muß sie eingefügt werden. Dies geschieht durch MVDOWN. Dabei wird der Platz für die neue Zeile gewonnen und anschließend wird mit MVUP die neue Zeile vom Textzwischenspeicher in das Programm einkopiert. Danach wird wieder auf den Zweig ST3 gesprungen und mit „>" angekündigt, daß die Zeile übernommen wurde und eine neue Eingabe erwartet wird.

Abb. 3.3.4-2 zeigt noch einmal die genauen Verhältnisse des Textspeichers und der einzelnen Pointer.

Einzelne Befehle

Da wäre zunächst die Zuweisung. Sie wurde schon vorher im Programmteil DEFLT angeschnitten. Falls kein CR auf der aktuellen Textposition (DE) steht, wird zur LET-Routine verzweigt, in die auch direkt durch Voranstellen des Basic-Befehls LET gelangt werden kann. Dort wird die Routine SETVAL aufgerufen, die die Zuweisung behandelt. Anschließend wird auf ein Komma abgefragt und, falls dieses vorhanden, erneut LET angesprungen. Das Syntaxdiagramm ist in *Abb. 3.3.4-3* dargestellt. SETVAL prüft, ob eine Varia-

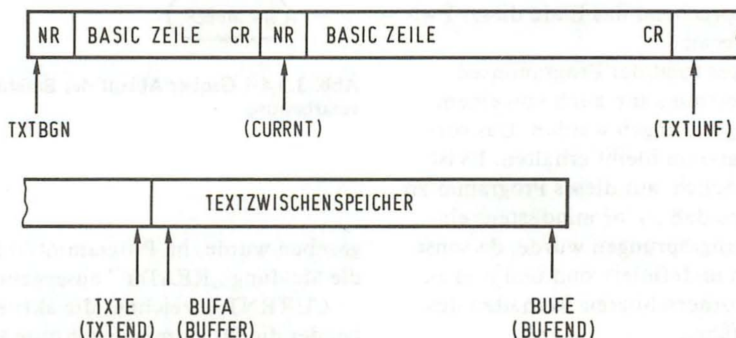


Abb. 3.3.4-2 Pointer im Textspeicher

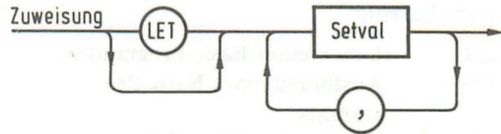
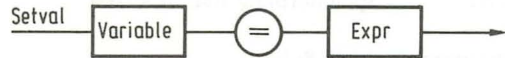


Abb. 3.3.4-3 Syntaxdiagramm der Zuweisung



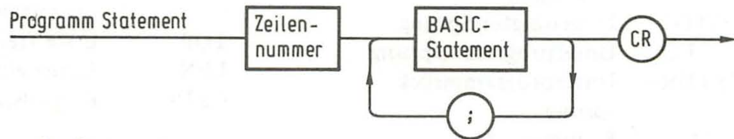


Abb. 3.3.4-4 Syntaxdiagramm des Statements

ble am Anfang, dann nachfolgend ein Gleichheitszeichen und schließlich ein arithmetischer Ausdruck steht. Der Wert dieses Ausdrucks wird dann der angegebenen Variablen zugewiesen.

Nach der Zuweisungssequenz wird zum Programmteil FINI gesprungen. Dort wird FIN aufgerufen und, falls das Programm zurückkehrt, zu der Fehlermeldung QWHAT gesprungen. *Abb. 3.3.4-4* verdeutlicht die Aufgabe von FINI. Eine Basic-Programmzeile besteht aus einer Zeilennummer sowie einem oder mehreren Befehlen, die durch je einen Strichpunkt zu trennen sind. Das Zeilenende ist durch ein Carriage Return (CR, Code 0DH) gekennzeichnet. In FIN wird überprüft, ob ein Strichpunkt folgt. Falls ja, so wird der Programmteil RUNSML ausgeführt, der die Aufgabe hat, die Ausführung des Programms in der gleichen Zeile durchzuführen. Folgte aber ein CR, so erfolgt der Sprung auf RUNNXL, wo die nächste Zeile ausgeführt wird, falls eine solche vorhanden ist. Im Fehlerfall erfolgt ein Rücksprung.

Abb. 3.3.4-5 zeigt eine Gesamtübersicht über alle Befehle, die nun nach und nach besprochen werden sollen. Dabei wurden die Funktionen und Vergleichsbefehle schon im vorhergehenden Kapitel abgehandelt.

Zuvor noch einige Beispiele zur Anweisung LET: Folgendes Programm ist syntaktisch möglich:

```
100  LET A=1
110  B=A+3*(A-1)
120  C=A=B+1
130  D=(1=A)+(B=7)
140  @(A)=@(A)+1;A=2
```

A erhält den Wert 1, B den Wert 1, C wird der Wert des Vergleichs A=B+1 zugewiesen. Dieser Ausdruck ist falsch, also erhält C den Wert 0. D erhält den Wert 1, da der Ausdruck 1=A wahr ist und den Wert 1 erhält, B=7 ist falsch und erhält den Wert 0. 1+0 ergibt 1 somit ist D=1.

Direkt-Befehle:

LIST	Listen eines Basic-Programms
RUN	Ausführen eines Basic-Programms
NEW	Löschen des Basic-Programms
BYE	Beenden des Basics
END	Speicherplatz neu besetzen

Programmierbare Befehle:

NEXT	Schleifenende
LET	Anfang einer arithmetischen Zuweisung
IF	Bedingung
GOTO	Unbedingter Sprung
GOSUP	Unterprogrammsprung
RETURN	Unterprogrammrücksprung
REM	Kommentar
FOR	Schleifenanfang
INPUT	Eingabe
PRINT	Ausgabe
STOP	Beenden des Ablaufs
CALL	Unterprogrammaufruf – Maschinenprogramm
OUTCHAR	Einzelzeichenausgabe
OUT	Ausgabe an Port
O$	Ausgabe eines Textes
I$	Eingabe eines Textes
POKE	Direkter Speicherzugriff
TAB	Tabulator ausgeben
BYTE	Sedezimalausgabe in Byteform
WORD	Sedezimalausgabe in Wortform

Funktionen:

RND	Zufallswert
ABS	Absolutbetrag
SIZE	Speicherfreiraum
PEEK	Direktspeicherzugriff
INCHAR	Einzelzeicheneingabe
HEX	Umwandlung vom Sedezimalsystem
IN	Porteingabe
' '	Einzelzeichenumwandlung
TOP	Erste freie Speicherzelle
LEN	Länge eines Strings
CSTS	Konsolstatus

Divers:

TO	In Schleifenanweisung
STEP	In Schleifenanweisung

Vergleichsbefehle:

>=
#
>
=
<=
<

Abb. 3.3.4-5 Gesamtübersicht über alle Befehle

In Zeile 140 wird das eindimensionale Feld verwendet. Dabei wird der Inhalt der ersten Elemente dieses Feldes um 1 erhöht und dem Feld wieder zugewiesen. Außerdem wird anschließend der Variablen A der Wert 2 zugewiesen.

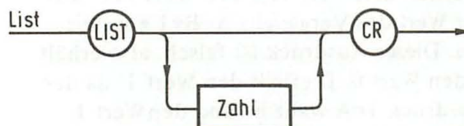


Abb. 3.3.4-6 Syntaxdiagramm von LIST

LIST

Es ist also schon möglich, ein Programm einzugeben. Das Programm soll aber auch zur Kontrolle wieder ausgegeben werden können. Dazu wird der Befehl LIST verwendet; *Abb. 3.3.4-6* zeigt das dazugehörige Syntaxdiagramm. LIST kann demnach allein stehen oder von einer Zahl gefolgt werden. Ob eine Zahl vorhanden ist, wird mit dem Aufruf TSTNUM geprüft. Anschließend folgt ein Test, ob danach ein CR steht. Falls nein, wird eine Fehlermeldung ausgegeben. Vor dem Aufruf von FNDLN ist als Zeilennummer 0 vorhanden,

wenn keine zusätzliche Zahl eingegeben wurde, sonst diese Zahl. FNDLN sucht nach dieser Zeile. Wurde keine gefunden, so wird ein Carry übergeben und nach RSTART verzweigt. Sonst wird mit PRTLN die Zeile ausgegeben und dann durch den Aufruf CONT überprüft, ob ein CTRL-C auf der Konsole eingegeben wurde und falls ja, wird in dem Unterprogramm auf RSTART gesprungen. Falls nicht, wird mit FNDLP die nächste Zeile gesucht und dies geschieht dann solange, bis keine Zeile mehr da ist, die ausgegeben werden muß.

Der direkte Sprung aus dem Unterprogramm CONT ist möglich, da in dem angesprungenen Programm RSTART eine Korrektur des Stackpointers vorgenommen wird. Wäre dies nicht der Fall, so würde der Stack immer weiter nach unten laufen, bis er mit Variablengebieten zusammenstößt.

Beispiele:

LISTcr	gibt das gesamte Basic-Programm aus
LIST 110	gibt das Basic-Programm beginnend mit Zeile 110 aus
LISTcr CTRL-C	gibt das Programm solange aus, bis CTRL-C (Code 03) eingegeben wurde.

RUN

Der wohl wichtigste Programmteil dient zum Starten des Basic-Programms. Dazu wird erst geprüft, ob RUN mit einem CR abgeschlossen wurde, und danach das Registerpaar DE mit der Adresse TXTBGN belegt. Dadurch wird erreicht, daß die Programmausführung mit der ersten Zeile beginnt. Durch den Aufruf FNDLP wird die Suche nach der ersten Zeile veranlaßt. HL wurde dazu zuvor mit 0 belegt. Falls nach dem Aufruf ein Carry vorhanden ist, gibt es keine Zeile und RSTART wird ausgeführt.

Sonst wird CURRNT mit der aktuellen Zeile belegt und RUNSML aufgerufen. In RUNSML erfolgt mit dem Aufruf von CONT als erstes eine Überprüfung auf CTRL-C. Falls dieses von der Konsole eingegeben wurde, so wird RSTART angesprungen und damit die Ausführung beendet. Sonst wird mit Hilfe von EXEC nach einem zulässigen Basic-Befehl, beginnend bei Tabelle TAB2,gesuchtund ein entsprechendes Programm von dort aus gestartet.

NEW

NEW löscht ein vorhandenes Basic-Programm. Es wird dabei einfach TXTBGN in die Variable TXTUNF gespeichert. Anschließend wird STOP ausgeführt und dabei auf RSTART gesprungen.

BYE

BYE beendet die Ausführung des Basic-Interpreters durch einen RST-7-Befehl. Hier kann auch leicht ein anderer Befehl verwendet werden, der zum Beispiel direkt in einen Monitor springt. RST 7 wurde gewählt, da bei dem hier verwendeten Monitor mit RST 7 ein „Break" ausgeführt wird.

END

END wird normalerweise dazu verwendet, um das Programmende zu kennzeichnen. Dies wurde hier aber nicht getan, da das Ende eines Basic-Programms automatisch durch den letzten Befehl definiert ist. Damit steht der Befehl END wieder frei. END wird hier deshalb verwendet, um dynamisch den für ein Basic-Programm zur Verfügung stehenden Platz zu vergrößern. Dies wirkt dann genauso, als ob die im Listing stehende Anweisung DS 600 hinter TXTBGN: vergrößert würde.

Um dies zu erreichen, erhält END einen Parameter, der im Programm durch den Befehl „CALL EXPR" ins HL-Register gebracht wird. Danach folgt eine Bereichsüberprüfung, um zu verhindern, daß der END-Befehl zu einem Zusammensturz führt, wenn zum Beispiel das neue Ende

vor das Programm gelegt wird, oder auf einen nicht vorhandenen Speicherplatz. In diesem Fall wird die Fehlermeldung „SORRY" ausgegeben, die immer wenn sie auftritt bedeutet, daß eine Forderung an den Interpreter syntaktisch möglich wäre, aber unter den momentanen Voraussetzungen nicht erfüllbar ist. War der Wert zulässig, werden BUFEND, BUFFER und TXTEND neu belegt. Für den Textbuffer sind nun 132 Zeichen zugelassen, so daß längere Eingabezeilen angegeben werden können. Anschließend wird auf RSTART gesprungen.

GOTO
Bei der Ausführung dieses Befehls wird mit dem Aufruf CALL EXPR ein Parameter geholt, der die Zeilennummer angibt, auf die gesprungen werden soll. Die Besonderheit liegt hier darin, daß auch arithmetische Ausdrücke als Parameter zulässig sind. Damit können berechnete Sprünge durchgeführt werden und da in arithmetischen Ausdrücken auch Vergleiche zulässig sind, können auch bedingte Sprünge und Mehrfachverzweigungen realisiert werden.

Der in HL stehende Wert der gesuchten Zeilennummer wird an die Routine FNDLN gegeben, die die Zeile sucht. Wird sie nicht gefunden, so wird AHOW aufgerufen und eine Fehlermeldung ausgegeben.

Anschließend wird RUNTSL aufgerufen, wenn kein Fehler vorlag und dort die Programmausführung bei der neuen Zeile begonnen. Nach dem Aufruf von EXPR wird das Registerpaar DE gerettet, um im Fehlerfall für AHOW die Position des Fehlers festzuhalten. Mit POP PSW wird, falls kein Fehler auftrat, DE einfach weggeworfen.

GOSUB
GOSUB bewirkt einen Unterprogrammaufruf. Es werden zuerst alle Parameter eines eventuellen Schleifenaufrufs gerettet. Dann wird die Sprungadresse geholt (wie bei GOTO ebenfalls ein arithmetischer Ausdruck) und mit PUSH DE der Textzeiger gerettet, so daß danach wieder an der alten Stelle das Basic-Programm fortgesetzt werden kann, wenn ein RETURN-Befehl erfolgte.

Beispiele für GOTO und GOSUB

```
100 GOTO 120
120 GOTO (A=1)*100+(A=2)*130
130 GOSUB 500
...
```

In Zeile 100 wird nach 120 gesprungen und in Zeile 120 wieder nach Zeile 100, falls A=1 ist, oder auf Zeile 130, falls A=2 ist. Ist A weder 1 noch 2, erfolgt eine Fehlermeldung. In 130 wird ein Unterprogrammaufruf auf Zeile 500 durchgeführt.

RETURN
Um den GOSUB-Befehl zu vervollständigen, wird noch der RETURN-Befehl benötigt. Es werden dort in umgekehrter Reihenfolge alle Programmparameter vom Stack zurückgeholt. Durch den Aufruf FINI wird erreicht, daß in der alten Zeile genau hinter dem GOSUB-Befehl mit der Ausführung des Basic-Programms fortgefahren wird.

IF
Die Ausführung von IF beginnt bei der Marke IFF. Dort wird zunächst durch den Aufruf EXPR ein arithmetischer Ausdruck geholt; natürlich normalerweise aus einem Vergleich bestehend. Ist der Ausdruck nicht 0, so wird in der gleichen Zeile fortgefahren. Dies geschieht durch den Aufruf von RUNSML. Ist der Ausdruck gleich 0, der Vergleich also nicht erfüllt, dann wird mit FNDSKP der Rest der Zeile übersprungen und, falls eine neue Zeile da ist, diese mit RUNTSL ausgeführt, sonst RSTART.

Beispiel:

```
100 IF A=1 GOTO 300
110 IF B<=(C*D) A=2
120 B=100
300 C=2, D=3;A=200
```

Falls A am Anfang den Wert 1 besaß, wird auf 300 gesprungen und dort C mit 2, D mit 3 und A mit 200 belegt. War A nicht1, so wird A mit 2 belegt, falls „B Kleiner gleich C mal D" war und dann in jedem Fall B mit 100 belegt und schließlich ebenfalls Zeile 300 ausgeführt.

REM

REM wirkt wie eine IF-Anweisung, die immer das Ergebnis „falsch", also den Wert 0 ergibt. Damit ist es möglich, Kommentare in das Programm aufzunehmen. HL wird dazu mit 0 belegt und dann auf IFFR gesprungen. Durch dieses etwas eigentümliche Verfahren wird Programmspeicherplatz gespart.

FOR

Abb. 3.3.4-7 zeigt das Syntaxdiagramm zur FOR-Schleife. Mit PUSHA werden zunächst die alten Parameter auf den Stack gerettet. Dadurch ist es möglich, FOR-Schleifen zu schachteln. Anschließend wird durch den Aufruf von SETVAL die Zuweisung durchgeführt. Die Adresse der Zuweisungsvariablen wird in LOPVAR gespeichert.

Mit Hilfe des EXEC-Moduls wird getestet, ob nun das Wort TO folgt. Falls nein, wird die Fehlermeldung WHAT ausgegeben, sonst wird FR1 angesprungen. Dort wird das Ende der Schleife durch den Aufruf von EXPR geholt und in LOPLMT abgespeichert – diesmal der Wert selbst, nicht eine Adresse. Wird bei der FOR-NEXT-Schleife der Begriff STEP weggelassen, so wird automatisch eine Schrittweite von 1 angenommen. Dies wird hier durch den Aufruf von EXEC (beginnend bei TAB6) erreicht. Ist STEP nicht vorhanden, so wird nach FR3 gesprungen, sonst auf FR2. Bei FR3 wird HL mit 1 belegt, auf FR2 wird mit EXPR ein Wert geholt. Anschließend wird auf FR4 der Wert in LOPINC gespeichert. Die aktuelle Zeilennummer wird in LOPLN gespeichert und der Textpointer in LOPPT.

Als nächstes wird untersucht, ob in einer alten FOR-Schleife, die noch auf dem Stack ist, die gleiche Schleifenvariable verwendet wurde. Falls ja, so werden alle Parameter dieser Schleife aus dem Stack mit dem MVDOWN-Programm entfernt. Ein Parameterblock hat dabei eine Länge von 10. Bei FR8 wird die Ausführung fortgesetzt.

NEXT

Es wird zunächst geprüft, ob NEXT von einer Variablen gefolgt wird. Falls nein, so wird QWHAT aufgerufen und eine Fehlermeldung ausgegeben; sonst wird die Adresse der Variablen in VARNXT zwischengespeichert. Es wird nun LOPVAR geladen und geprüft, ob der Wert 0 ist. Falls ja, so wurde keine FOR-Anweisung gegeben, und es wird ein Fehler ausgegeben.

Es werden dann die beiden Adressen verglichen. Falls sie übereinstimmen, wird auf NX 3 gesprungen, sonst wird mit POPA ein Schleifenblock vom Stack entfernt und erneut untersucht. Bei NX3 wird die Schleifenvariable um LOPINC erhöht. Dann wird überprüft, ob LOPLMT schon erreicht wur-

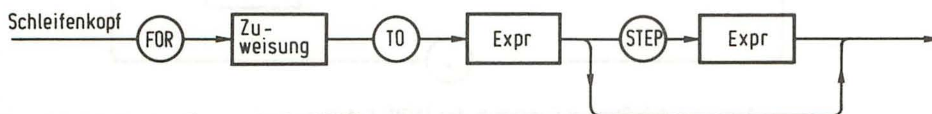


Abb. 3.3.4-7 Syntaxdiagramm der FOR-Anweisung

de. Falls ja, wird mit POPA der Stack geleert und die Programmausführung mit der folgenden Anweisung fortgesetzt. Falls die Grenze nicht erreicht wurde, so wird die Zeilennummer der FOR-Schleife und der Textpointer nach der FOR-Anweisung geladen und an dieser Stelle mit der Programmausführung fortgefahren.

Beispiele:

```
100 FOR I=1 TO 10
.
.
.
120 NEXT I
130 FOR N=10 TO 2 STEP -2
140 FOR J=1 TO 5 STEP 3
150 NEXT J
160 NEXT N
```

Die zwischen 100 und 120 liegenden Anweisungen werden 10 mal durchlaufen. Bei 130 beginnt eine verschachtelte Schleife. Die äußere Schleife wird, beginnend mit 10, um jeweils zwei mit der Schleifenvariable N auf 2 zurückgezählt; die innere Schleife zählt aufwärts in dreier Schritten von 1 ab, bis 5 überschritten wird.

INPUT

Mit dieser Anweisung ist es möglich, numerische Variable einzugeben. *Abb. 3.3.4-8* zeigt das dazugehörige Syntaxdiagramm. Bei IP1 wird zuerst geprüft, ob ein String angegeben wurde. Dies geschieht durch QTSTG, der den String auch gleich ausgibt. Falls es ein String war, so werden die nach dem Aufruf stehenden drei Bytes übersprungen. Sonst wird IP2 angesprungen.

Falls ein String vorkam, wird geprüft, ob nachfolgend eine Variable steht; falls ja, wird auf IP3 gesprungen, falls nein, wird bei IP4 geprüft, ob ein Komma folgt und danach wieder auf IP1 gesprungen, sonst INPUT beendet. Bei IP2 wird der Variablenname als Text ausgegeben, so daß sofort erkennbar ist, welche Variable eingegeben werden muß. War ein String vorhanden, so geschieht dies nicht, da gleich auf IP3 gesprungen wird. Dort wird nun zunächst gesichert, daß im Fehlerfall die Routine INPERR angesprungen wird. Dies geschieht durch die Belegung von CURRNT mit einer Adresse IP1, die auf einen negativen Wert (D5H) zeigt. Dieser Wert wird in der ERROR-Routine abgeprüft.

Lag kein Fehler vor, so wird der Wert, der durch EXPR vom Benutzer eingegeben wurde, in die Variable abgespeichert. Die Besonderheit liegt darin, daß hier bei INPUT zur Laufzeit auch die Eingabe von arithmetischen Ausdrücken möglich ist. Die Benutzereingabe wurde durch GETLN in BUFFER abgelegt. Bei IP4 wird geprüft, ob weitere Eingaben erfolgen sollen.

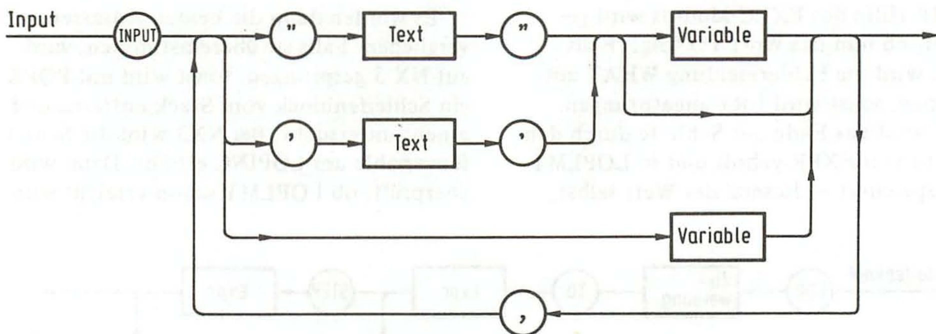


Abb. 3.3.4-8 Syntaxdiagramm der INPUT-Anweisung

Beispiele:
100 INPUT A
110 INPUT "TEXT"B,C
120 INPUT "TEXT",D

Bei der Ausführung erscheinen nacheinander folgende Ausgaben:
A:
TEXT:
C:
TEXTD:

Nach jedem Doppelpunkt wird eine Eingabe erwartet.

PRINT

Abb. 3.3.4-9 zeigt das Syntaxdiagramm von PRINT. Zunächst wird das Register C mit 6 geladen. Der Inhalt dieses Registers gibt die Länge an, mit der numerische Werte ausgegeben werden. Dabei werden Zahlen zu dieser Länge mit Leerräumen aufgefüllt. Somit ist eine bescheidene Art von formatierter Ausgabe möglich, da sich das Register C verändern läßt. Dies geschieht durch das Zeichen „#", das von einem arithmetischen Ausdruck gefolgt wird. Zuvor wird jedoch im Programmteil PRINT abgefragt, ob die Anweisung beendet ist, dadurch, daß ein CR oder ein Strichpunkt folgt. Falls ja, wird CRLF ausgegeben und der restliche Teil auf die übliche Weise ausgeführt.

Bei PRO erfolgt die Abfrage auf die Formatanweisung. Bei PR1 wird geprüft, ob vielleicht ein String ausgegeben werden soll. Ebenfalls in dem Unterprogramm QTSTG wird auf das Zeichen 5FH (Unterstreichung) abgefragt; wird dieses eingegeben, so wird 8DH ausgegeben, was einem CR mit gesetztem Paritätsbit entspricht. Dadurch wird in der Ausgabe verhindert, daß ein Line-Feed-Zeichen (LF) mit ausgegeben wird, und es kann bei einem druckenden Ausgabegerät eine Zeile mehrfach gedruckt werden, z.B. für Hervorhebungen.

War QTSTG negativ, so wird PR8 ausgeführt und dort ein arithmetischer Ausdruck ausgewertet. Der Wert wird dann durch PRTNUM mit der im C-Register gespeicherten Maximallänge ausgegeben und ggf. aufgefüllt. Durch Komma getrennt können mehrere solcher Ausdrücke in der PRINT-Anweisung verwendet werden. Hört

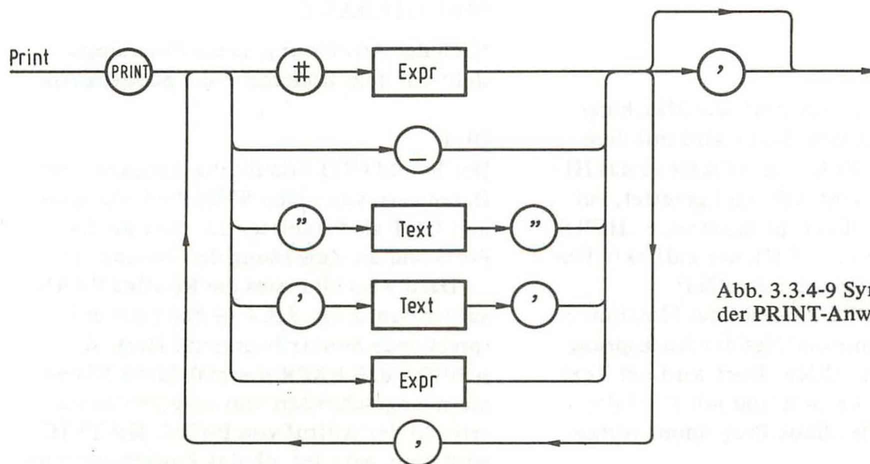


Abb. 3.3.4-9 Syntaxdiagramm der PRINT-Anweisung

PRINT mit einem Komma auf, so wird kein CRLF ausgegeben.

Beispiele:
50 A=2
100 PRINT "TEST TEXT",
110 PRINT A,#8,A,"TEXT"
120 PRINT
130 PRINT "A C E",–, "B D"

Bei der Ausführung wird folgendes gedruckt:

TESTbTEXTbbbbb2bbbbbbb2TEXT

(b steht für Blank)

cr

(cr nur Carriage return)

ABCDE

STOP

STOP beendet den aktuellen Programmlauf. Mit ENDCHK wird geprüft, ob die Anweisung mit CR beendet wird. Anschließend wird RSTART aufgerufen.

Beispiel:
100 A=1
110 STOP
120 A=2

Nach Ausführung dieses Programms hat A den Wert 1.

CALL

CALL dient dem Aufruf von Maschinen-Unterprogrammen. Dazu wird mit dem Aufruf von EXPR eine Adresse in das HL-Register gebracht. DE wird gerettet, auf den Stack die Rücksprungadresse „HERE" gebracht und mit PCHL der indirekte Unterprogrammsprung ausgeführt.

Nach dem RET-Befehl des Maschinen-Unterprogramms erfolgt der Rücksprung auf die Marke HERE. Dort wird der Textpointer zurückgeholt und mit FINI die Ausführung des Basic-Programms fortgesetzt.

Beispiel:
100 CALL HEX (4000)
. . .

```
4000H: LDA 5000H
       INR A
       STA 5000H
       RET
5000H: Zählerstand
```

Durch den Aufruf in Zeile 100 wird der Inhalt der Zelle 5000H um eins erhöht. Auf diesen Wert kann dann zum Beispiel mit PEEK zugegriffen werden.

OUTCHAR

Dieser Befehl gehört nicht zum Standard-Basic und dient der Ausgabe eines einzelnen Zeichens. Mit EXPR wird ein Wert geholt, der im ASCII-Code als Zeichen ausgegeben wird. Dazu wird der niederwertige Teil aus dem HL-Register in den Akkumulator geladen und mit OUTC ausgegeben. Der Aufruf von FINI führt das Programm fort.

Beispiel:
200 OUTCHAR 65
210 B=66
220 OUTCHAR B
230 C='C'
240 OUTCHAR C

Nach der Ausführung dieses Programms steht auf dem Bildschirm die Sequenz ABC.

OUT

Der Befehl OUT erlaubt die Ausgabe eines Datenwertes an einen 8080 Port. Dazu erhält OUT als Parameter die Adresse des Ports und als Zuweisung den Datenwert.

Dazu wird als erstes die Routine PARN aufgerufen. *Abb. 3.3.4-10* zeigt das entsprechende Syntaxdiagramm. Dort ist sichtbar, daß EXPR diesmal durch Klammern eingeschlossen sein muß; genau dies erreicht der Aufruf von PARN. Mit TSTC wird dann getestet, ob das Zuweisungssym-

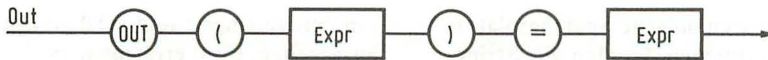


Abb. 3.3.4-10 Syntaxdiagramm von OUT

bol „=" folgt, und dann durch einen erneuten Aufruf von EXPR der Datenwert geholt. Als nächstes wird die Befehlssequenz D3 xx C9, beginnend bei IOBUFA, gespeichert, wobei für xx die Adresse des I/O-Ports eingesetzt wird, die durch den ersten Aufruf von EXPR geholt wurde. Diese selbstmodifizierende Technik ist nötig, da im 8080-Befehlssatz kein I/O-Befehl vorgesehen ist, der eine indirekte Adressierung erlaubt (im Gegensatz zum Z80, bei dem solche Befehle existieren).

Beispiel:
100 OUT(HEX(10))=20

Auf den I/O-Port mit der sedezimalen Adresse 10H(dez.16) wird der dezimale Wert 20 gegeben.

O$

Da eine Stringverarbeitung in einem Standard-Basic-Interpreter eine relativ umfangreiche Angelegenheit ist, wird hier eine stark vereinfachte Version durchgeführt, die aber durchaus sehr leistungsfähig ist. Bei der Marke „O:" beginnt das Programm.

O$ hat die Aufgabe, einen String auszugeben. Dabei steht dieser String an einer beliebigen Stelle im Speicher, möglichst hinter dem Basic-Programm. Der String wird durch ein Endekennzeichen begrenzt, wobei hier die binäre Null verwendet ist. O$ erhält als Parameter die Anfangsadresse des Strings. Dazu wird im Programm EXPR aufgerufen und die Adresse in das DE-Register transportiert. Damit kann durch die Routine PRTSTG der String ausgegeben werden. Durch den vorangestellten Befehl „XRA A" wird erreicht, daß die Ausgabe des Strings mit dem Zeichen NUL (binäre Null) beendet wird. Der Textpointer DE wird während dieses Vorgangs zwischengespeichert, um danach wieder für die normale Basic-Ausführung zur Verfügung zu stehen. Mit FINI wird der Programmlauf fortgesetzt.

Beispiel:
100 O$ 4096

Beginnend mit der Zelle 4096 wird ein dort stehender Text ausgegeben, bis der Wert NUL vorkommt, d.h. bis zu einem Null-Byte.

I$

Diese Anweisung dient der Eingabe eines Strings an eine als Parameter angegebene Speicheradresse. Dabei wird das Ende des Strings automatisch durch NUL gekennzeichnet. Durch den Aufruf von EXPR wird die Adresse geholt. Mit GL1 wird eine Zeile vom Benutzer angefordert, die durch die Eingabe eines CRs beendet wird. Das Zeichen CR wird selbst nicht mit abgespeichert, stattdessen das Zeichen NUL. Der Transport geschieht dazu zuvor mit MVUP. Anschließend wird noch die Variable LEGT mit der Länge des Strings belegt, so daß beim nächsten Aufruf von LEN die richtige Länge vorbesetzt ist.

Beispiel:
100 I$ TOP
110 O$ TOP+1

Wurde bei der Eingabe ABCD angegeben, so wird bei der Ausgabe dieses Textes BCD ausgegeben. TOP ist die Variable für den ersten freien Speicherplatz hinter dem Basic-Programm und kann dafür verwendet

werden, eine dynamische Speicherplatzzuweisung zu`erreichen. Werden alle Stringvariable durch Bezug auf TOP gewonnen, also TOP,TOP+10,TOP+30, so wird immer die optimale Speicherplatzausnützung gesichert. Dabei muß darauf geachtet werden, daß keine Kollision mit dem eindimensionalen Feld auftritt.

POKE
Mit POKE kann eine Speicherzelle verändert werden. Dazu erhält POKE zwei Parameter, die Adresse und den Datenwert. *Abb. 3.3.4-11* zeigt das Syntaxdiagramm zu POKE. Die beiden Parameter werden durch Komma getrennt. Dazu wird die Routine TSTC aufgerufen. Der niederwertige Teil des Datenwertes wird an die Stelle, die der erste Parameter angibt, gespeichert.

Beispiel:
100 POKE 4096,10

Der Wert 10 wird auf die Adresse 4096 gespeichert.

TAB
TAB, dessen Syntax *Abb. 3.3.4-12* zeigt, wird hier als eigener Befehl verwendet und nicht, wie sonst oft üblich, als „SPC" im PRINT-Befehl. TAB erhält als Parameter in Klammern einen arithmetischen Ausdruck, der die Anzahl der Blanks (Leerzeichen) angibt, die ausgegeben werden sollen. Ist der Wert 0, so werden keine Leerzeichen ausgegeben, sonst wird der Wert um eins verringert und solange Leerzeichen ausgegeben, bis 0 erreicht wird.

Beispiel:
200 PRINT "A",
210 TAB(10)
220 PRINT "E"

Bei der Ausführung erscheint folgendes:

AbbbbbbbbbbE

wobei b für ein Blank steht.

BYTE
BYTE gibt den Wert, der als dezimaler Parameter hinzugefügt wurde, in sedezimaler Schreibweise aus. Dabei werden nur dezimale Zahlen bis 255 in Form zweier sedezimaler Stellen ausgegeben.

Beispiel:
100 BYTE(17)

ergibt den Ausdruck 11.

WORD
WORD arbeitet wie BYTE, nur werden vier sedezimale Stellen ausgegeben. Die Unterprogramme WRIT2 und IST dienen der Ausgabekonvertierung.

Beispiel:
100 WORD(1023)

ergibt den Ausdruck 03FF

Dabei wird bei WORD wie auch bei BYTE kein CRLF ausgegeben, so daß die Ausgabeformatierung durch PRINT-Befehle gesteuert werden kann.



Abb. 3.3.4-11 Syntaxdiagramm von POKE



Abb. 3.3.4-12 Syntaxdiagramm von TAB

3.3.5 Basic-Programmbeispiele

Abb. 3.3.5-1 zeigt einige Beispiele mit dem RDK-BASIC. Es muß noch erwähnt werden, daß diese Basic-Programme wegen einiger vorher bereits besprochener Syntax-Unterschiede nicht ohne Änderung auf Standard-Basic-Interpretern laufen (PET, CBM, TRS-80, Apple usw.). Die nötigen Anpassungen lassen sich aber leicht durch Vergleich der Systemhandbücher mit dem vorstehenden Kapitel herausfinden.

```
  10 REM STRINGVERARBEITUNG
  20 PRINT "EINGABE EINES STRINGS :",
  30 I$ TOP;REM EINGABE DES STRINGS
  40 S=0
  50 FORI=TOP TO TOP+LEN-2
  60 IFPEEK(I)>PEEK(I+1)A=PEEK(I+1);POKEI+1,PEEK(I);POKEI,A;S=1
  70 NEXTI
  80 IF S=1 GOTO 40
  90 PRINT
 100 PRINT"SORTIERTE BUCHSTABEN DES STRINGS"
 110 O$ TOP

READY
>RUN
EINGABE EINES STRINGS :STRING ZUM SORTIEREN.

SORTIERTE BUCHSTABEN DES STRINGS
   .EEGIIMNNORRRSSTTUZ
READY
>
```

```
  10 REM ARRAYVERARBEITUNG
  20 FOR I=1 TO 10
  30 @(I)=0; REM LOESCHEN
  40 NEXT I
  50 FOR I=1 TO 10
  60 FOR J=1 TO 10
  70 @(I)=@(I)+RND(100)
  80 NEXT J
  90 NEXT I
 100 FOR I=1 TO 10
 110 PRINT #4,@(I),
 120 NEXT I
 130 PRINT

READY
>RUN
 492 451 530 617 506 699 590 537 659 505

READY
>RUN
 482 375 422 313 507 544 371 508 510 512

READY
>RUN
 552 301 413 583 511 489 545 499 353 587
```

Abb. 3.3.5-1 Beispiele für das RDK-BASIC

```
 10 REM VERGLEICHE
 20 PRINT
 30 INPUT"WERT FUER A"A,"WERT FUER B"B
 40 PRINT #3,A,B,A=B,A<B,A>B,A<=B,A>=B,A#B
 50 GOTO 30

READY
>RUN

WERT FUER A:2
WERT FUER B:3
  2  3  0  1  0  1  0  1
WERT FUER A:3
WERT FUER B:2
  3  2  0  0  1  0  1  1
WERT FUER A:2
WERT FUER B:2
  2  2  1  0  0  1  1  0
WERT FUER A:-1
WERT FUER B:-3
 -1 -3  0  0  1  0  1  1
WERT FUER A:-3
WERT FUER B:-1
 -3 -1  0  1  0  1  0  1
WERT FUER A:-3
WERT FUER B:-3
 -3 -3  1  0  0  1  1  0
WERT FUER A:
WHAT?
WERT FUER A:A+1
WERT FUER B:B=3
 -2  0  0  1  0  1  0  1
WERT FUER A:A+1
WERT FUER B:B-1
 -1 -1  1  0  0  1  1  0
WERT FUER A:2
WERT FUER B:3
  2  3  0  1  0  1  0  1
WERT FUER A:A+1
WERT FUER B:B-4
  3 -1  0  0  1  0  1  1
WERT FUER A:RND(20)
WERT FUER B:RND(20)
 14 10  0  0  1  0  1  1
WERT FUER A:HEX(10)
WERT FUER B:TOP-TOP+5
 16  5  0  0  1  0  1  1
WERT FUER A:@(5)
WERT FUER B:@(4)
511563  0  1  0  1  0  1
```

Abb. 3.3.5-1

```
 10 REM ARITHMETIK MIT RELATIONEN
 20 INPUT"1=TEST1 2=TEST2"T
 30 GOTO (T=1)*100+(T=2)*200+(T<1)*20+(T>2)*20
100 PRINT"TEST1";GOTO 20
200 PRINT "TEST2";GOTO 20

READY
>RUN
1=TEST1 2=TEST2:1
TEST1
1=TEST1 2=TEST2:2
TEST2
1=TEST1 2=TEST2:5
1=TEST1 2=TEST2:4
1=TEST1 2=TEST2:3
1=TEST1 2=TEST2:-2
1=TEST1 2=TEST2:1
TEST1
1=TEST1 2=TEST2:5
1=TEST1 2=TEST2:3-2
TEST1
1=TEST1 2=TEST2:5-4+1
TEST2
1=TEST1 2=TEST2:
```

Abb. 3.3.5-1

```
READY
>PRINT TOP,SIZE
  4551   452

READY
>END HEX(3FFF)-140

READY
>PRINT TOP,SIZE
  4551 11559

READY
>WORD(TOP)
11C7
```

```
 10 REM FEHLERBEHANDLUNG
 20 PRINT 2+4,5,555*888,4,8

READY
>RUN
     6      5HOW?
 20 PRINT 2+4,5,555*888?,4,8
```

```
>10 REM ANWENDUNG VON INCHAR UND OUTCHAR
>20 OUTCHAR(INCHAR+1);REM UMKODIERUNG
>30 GOTO 20
>RUN
UFTU!EFT!VNLPEJFSFOT

READY
>LIST
  10 REM ANWENDUNG VON INCHAR UND OUTCHAR
  20 OUTCHAR(INCHAR-1)
  30 GOTO 20
```

Abb. 3.3.5-1

```
  10 REM TAB FUNKTION
  20 FORI=-8 TO 8
  30 PRINT#2,I,"!",
  40 TAB(I*I)
  50 PRINT"*"
  60 NEXT I

READY
>RUN
-8!                                                                *
-7!                                                 *
-6!                                    *
-5!                         *
-4!                *
-3!         *
-2!    *
-1! *
 0!*
 1! *
 2!    *
 3!         *
 4!                *
 5!                         *
 6!                                    *
 7!                                                 *
 8!                                                                *

READY
>
```

```
 10 REM KLEINES MONITORPROGRAMM
 20 INPUT"BEFEHL 1=GOTO 2=DUMP"B
 30 GOTO (B=1)*100+(B=2)*200+(B<1)*20+(B>2)*20
100 INPUT"ADRESSE"A
110 CALL A
120 GOTO 20
200 INPUT"ANFADRESSE"A,"ENDADRESSE"E
210 FORI=ATOE STEP 8
220 WORD(I);PRINT"  ",
230 FORJ=0TO7
240 BYTE(PEEK(I+J));PRINT " ",
250 NEXTJ
260 PRINT
270 NEXTI
280 PRINT;GOTO 20
```

Abb. 3.3.5-1

```
READY
>RUN
BEFEHL 1=GOTO 2=DUMP:2
ANFADRESSE:HEX(500)
ENDADRESSE:HEX(50F)
0500  C9 0F 05 C3 D5 05 C3 03
0508  F0 C3 00 D0 C3 12 F0 31

BEFEHL 1=GOTO 2=DUMP:1
ADRESSE:HEX(F01E)
BEFEHL 1=GOTO 2=DUMP:##
```

3.3.6 Erweiterungsmöglichkeiten des Basic-Interpreters

3.3.6.1 Fließkomma-Arithmetik

Unser Basic-Interpreter beherrscht nur die Ganzzahl-Arithmetik (Integer). Dadurch ist er für wissenschaftliche Berechnungen meist nicht einsetzbar. Hier soll nun kurz aufgezeigt werden, wie es möglich ist, ein „Gleitkommapaket" aufzubauen. *Abb. 3.3.6.1-1* zeigt das Listing eines eingeschränkten „Floatingpoint-package". Es ist fähig, zwei Grundrechenarten (+ und –) auszuführen.

Bei einer Gleitkommaarithmetik wird eine Zahl durch eine Mantisse und einer „Charakteristik" dargestellt, z.B. .123 E 23. Dabei ist 123 die Mantisse und 23 der Exponent. Es müssen außerdem noch die Vorzeichen mit gespeichert werden. Dazu wird meist für die Mantisse das Zweierkomplement verwendet und beim Exponent eine Konstante aufaddiert, so daß dadurch die Charakteristik entsteht. *Abb. 3.3.6.1-2* zeigt die hier verwendete Darstellung.

Die Zahlendarstellung erfolgt wegen der Lesbarkeit im BCD-Format, so daß auch Umwandlungsroutinen von Binär in BCD und umgekehrt entfallen. Ein weiterer Vorteil einer BCD-Arithmetik ist die Genauigkeit, da Rundungsfehler beim Umwandeln entfallen. Die Charakteristik wird mit vier BCD-Stellen dargestellt, was 2 Bytes entspricht. Die Additionskonstante ist 5000. Diese Konstante ist bei Addition und Subtraktion nicht von Belang, sondern müßte erst bei Multiplikation und Division berücksichtigt werden. Die Mantisse wird mit zehn BCD-Stellen dargestellt und deren Vorzeichen mit einer zusätzlichen Stelle, die auch eine Überlaufstelle beinhaltet. Bei einem negativen Vorzeichen stehen an der Stelle 2 die Ziffern 99 und bei einer positiven Mantisse 00.

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                        PAGE 1
.MAIN. -


                                .PHEX
                                .PABS
  0100                          .LOC 100H
                                ;*************************
                                ;* BCD FLOATING POINT *
                                ;* PACKAGE 801110  RDK *
                                ;*************************
  0100      C3 F01E             JMP 0F01EH       ;MONITOR AUFRUF
  0103      C3 014B             JMP FLOATADD
  0106      C3 0187             JMP FLOATCOM

                                ;
                                ;
  0005                          LENM=5  ;MANTISSENLAENGE
  0002                          LENC=2  ;CHARACTERISTIK
  5000                          CHARC=5000H      ;DISPLACEMENT CHARACTERISTIK
  0006                          CHAR=6  ;POSITION CHARACTERISTIK
                                ;
                                ;AUFBAU
                                ;  __ __ __ __  __ __ __
                                ; CC CC VZ H   ...   L
                                ;          10 8  6  4  2
                                ;            9  7  5  3  1
                                ;VZ IST UEBERLAUF UND VORZEICHEN
                                ; 99 IST NEGATIVE ZAHL
                                ; 00 POSITIV
                                ; ANDERE ZAHELEN SIND UEBER- ODER UNTERLAEUFE
                                ;
                                ;
                                ; (HL)+(DE)->(HL) BCD ADD MANTISSE
                                ; IN B IST LAENGE ANZUGEBEN
  0109                          ADBCD:
  0109      E5                  PUSH H
  010A      D5                  PUSH D
  010B      AF                  XRA A
  010C                          ..LP:
  010C      1A                  LDAX D  ;AUSFUEHREN
  010D      8E                  ADC M
  010E      27                  DAA
  010F      77                  MOV M,A
  0110      2B                  DCX H
  0111      1B                  DCX D
  0112      10F8                DJNZ ..LP
  0114      D1                  POP D
  0115      E1                  POP H
  0116      C9                  RET
                                ;
                                ;
                                ; KOMPLEMENT   -(HL)->(HL) IN BCD
  0117                          COMBCD:
  0117      E5                  PUSH H
  0118      AF                  XRA A
  0119                          ..LP:
  0119      3E00                MVI A,0
  011B      9E                  SBB M
  011C      27                  DAA
  011D      77                  MOV M,A
```

Abb. 3.3.6.1-1 Listing des BCD-Fließkomma-Programmpakets

```
011E    2B              DCX H
011F    10F8            DJNZ ..LP
0121    E1              POP H
0122    C9              RET
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 2
.MAIN. -
```

```
                        ;
                        ;
                        ; RECHTSSCHIEBEN BCD
                        ; ARITHMETISCH !
0123                    RRBCD:
0123    E5              PUSH H
0124    D5              PUSH D
0125    58              MOV E,B
0126    1600            MVI D,0
0128    AF              XRA A
0129    ED52            DSBC D  ;B LAENGE+1 HL AUF LOW
012B    23              INX H
012C    7E              MOV A,M ;DABEI HL AUF VORZEICHENSTELLE
012D    0F              RRC
012E    0F              RRC
012F    0F              RRC
0130    0F              RRC
0131                    ..LP:
0131    ED67            RRD
0133    23              INX H
0134    10FB            DJNZ ..LP
0136    D1              POP D
0137    E1              POP H
0138    C9              RET
                        ;
                        ;
                        ;
                        ;
                        ;(HL)+1->(HL) BCD INCREMENT
                        ; B IST LAENGE
0139                    INRBCD:
0139    E5              PUSH H
013A    05              DCR B
013B    7E              MOV A,M
013C    C601            ADI 1
013E    27              DAA
013F    77              MOV M,A
0140    2B              DCX H
0141                    ..LP:
0141    7E              MOV A,M
0142    CE00            ACI 0
0144    27              DAA
0145    77              MOV M,A
0146    2B              DCX H
0147    10F8            DJNZ ..LP
0149    E1              POP H
014A    C9              RET
                        ;
                        ;
                        ; FLOAT ADD
                        ; ACCU1 IST DAT1
                        ; ACCU2 IST DAT2
                        ;(DAT1)+(DAT2)->(DAT1)
```

Abb. 3.3.6-1

```
014B                    FLOATADD:
014B    DD21 0207       LXI X,DAT1+7
014F    FD21 020F       LXI Y,DAT2+7

0153                    LOOPAN:
0153    DD7EF9          MOV A,-CHAR-1(X)           ;EXPONENT VERGLEICH
0156    FDBEF9          CMP -CHAR-1(Y)
```

TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21 PAGE 3
.MAIN. -

```
0159    381E            JRC D1ANGLEICH
015B    200A            JRNZ D2ANGLEICH
015D    DD7EFA          MOV A,-CHAR(X)
0160    FDBEFA          CMP -CHAR(Y)
0163    3814            JRC D1ANGLEICH
0165    281F            JRZ SKIPANGLEICH
0167                    D2ANGLEICH:
0167    21 020F         LXI H,DAT2+7
016A    0606            MVI B,LENM+1
016C    CD 0123         CALL RRBCD
016F    21 0209         LXI H,DAT2+7-CHAR
0172                    ONSO:
0172    0602            MVI B,LENC
0174    CD 0139         CALL INRBCD
0177    18DA            JMPR LOOPAN
0179                    D1ANGLEICH:
0179    21 0207         LXI H,DAT1+7
017C    0606            MVI B,LENM+1
017E    CD 0123         CALL RRBCD
0181    21 0201         LXI H,DAT1+7-CHAR
0184    18EC            JMPR ONSO

0186                    SKIPANGLEICH:
0186    21 0207         LXI H,DAT1+7      ;ADDIEREN MANTISSE
0189    11 020F         LXI D,DAT2+7
018C    0606            MVI B,LENM+1
018E    CD 0109         CALL ADBCD
0191                    NORM:
0191    DD21 0207       LXI X,DAT1+7
0195    DD7EFB          MOV A,1-CHAR(X)
0198    47              MOV B,A
0199    0F              RRC
019A    0F              RRC
019B    0F              RRC
019C    0F              RRC
019D    E60F            ANI 0FH
019F    4F              MOV C,A
01A0    78              MOV A,B
01A1    E60F            ANI 0FH
01A3    B9              CMP C    ;UELAUF=VZ DANN OK
01A4    2810            JRZ OKEQ
01A6    21 0207         LXI H,DAT1+7
01A9    0606            MVI B,LENM+1
01AB    CD 0123         CALL RRBCD
01AE    21 0201         LXI H,DAT1+7-CHAR
01B1    0602            MVI B,LENC
01B3    CD 0139         CALL INRBCD       ;SONST ANGLEICHEN
01B6                    OKEQ:
01B6    C9              RET
```

Abb. 3.3.6-1

```
                          ;
                          ;
                          ;
                          ; HIER KEIN CHECK OB EXPONENT UEBERLAEUFT.
                          ;FLOAT COMPLEMENT
                          ; DAT1 WIRD NEGIERT
 01B7                     FLOATCOM:
 01B7     21 0207         LXI H,DAT1+7
 01BA     0606            MVI B,LENM+1
 01BC     CD 0117         CALL COMBCD
 01BF     C3 0191         JMP NORM          ;GGF NORMIEREN
                          ;

TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 4
.MAIN. -


 0200                     .LOC 200H
 0200                     DAT1: .BLKB 8
 0208                     DAT2: .BLKB 8

                          .END
```

Abb. 3.3.6-1

```
DL Z80 CP/M DISK ASSEMBLER VERSION 2.21                     PAGE 5
.MAIN. -
+++++ SYMBOL TABLE +++++

ADBCD  0109         CHAR   0006           CHARC  5000
COMBCD 0117         D1ANGL 0179           D2ANGL 0167
DAT1   0200         DAT2   0208           FLOATA 014B
FLOATC 01B7         INRBCD 0139           LENC   0002
LENM   0005         LOOPAN 0153           NORM   0191
OKEQ   0186         ONSO   0172           RRBCD  0123
SKIPAN 0186         .BLNK. 0000:03 X      .DATA. 0000*  X
.PROG. 0000'   X
```

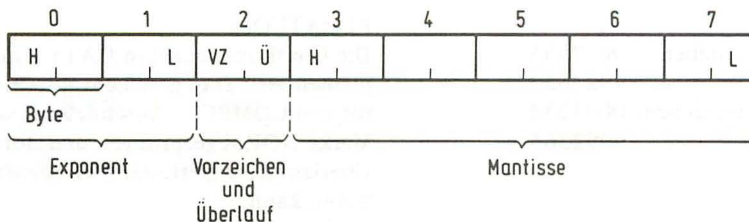


Abb. 3.3.6.1-2 Gleitkommadarstellung

Beschreibung der einzelnen Unterprogramme

ADBCD
Dieser Programmteil hat die Aufgabe, zwei BCD-Zahlen zu addieren. Dabei wird (HL) mit (DE) addiert und nach (HL) gespeichert. Die Länge der Zahlen wird im Register B angegeben. Dabei gibt B genaugenommen die Anzahl der Bytes an, wobei ein Byte aus zwei BCD-Ziffern besteht. Die eigentliche BCD-Addition wird durch den Befehl ADC M, DAA durchgeführt. Diese Routine wird verwendet, um die Mantissen zu addieren.

COMBCD
Hier wird eine BCD-Zahl, auf die HL zeigt, komplementiert. HL zeigt dabei auf die niederwertige Stelle der Zahl. Die Operation dafür lautet 1000...0 – x. Die Anzahl der Stellen wird im Register B angegeben.

RRBCD
Die Mantisse muß zur Korrektur rechtsverschoben werden. Dies entspricht einer Division durch 10. Das Vorzeichen muß dabei erhalten bleiben. Das Register B erhält dazu die Länge der Mantisse +1, um auch das Vorzeichen zu erfassen. Die gesamte Vorzeichenstelle wird mit verschoben, wobei von links nach rechts das Vorzeichen nachgeschoben wird. Die Verschiebung geschieht mit dem mächtigen Befehl RRD des Z80. Beim 8080 ist dies eine längere Sequenz von Befehlen.

Beispiel:

vor dem Verschieben:	0012345
	9923653
nach dem Verschieben:	0001234
	9992365

INRBCD
Das Programm dient der Erhöhung des Exponenten. Da es allgemein gehalten ist, wird in HL die Adresse des niederwertigen Teils angegeben und B gibt die Länge an (bei uns 2).

FLOATADD
Dies ist das eigentliche Additionsprogramm für eine Gleitkommaaddition. Es wird der Inhalt von DAT1 auf DAT2 addiert und das Ergebnis bei DAT1 gespeichert. DAT1 und DAT2 beinhalten jeweils eine Gleitkommazahl mit der Darstellung nach Abb. 2.2.6.1-2.

Als erstes wird untersucht, ob die beiden Charakteristiken gleich sind. Wenn ja, so wird auf SKIPANGLEICH gesprungen. Sind sie verschieden, so muß der Kleinere von beiden zuvor angeglichen werden. Dies geschieht bei D1ANGLEICH und D2ANGLEICH. Es wird jeweils die Mantisse nach rechts verschoben (Division durch 10) und die Charakteristik um eins erhöht. Dies geschieht solange, bis beide Charakteristika gleich sind. Hier könnte eine Programmoptimierung vorgenommen werden, die überprüft, ob mehr als 10 mal verschoben wurde, da dann die Mantisse den Wert 0 angenommen hat und ein weiteres Schieben keine Änderung bewirkt. Bei SKIPANGLEICH kann die Addition durchgeführt werden. Dabei kann ein Überlauf auftreten; dies wird anschließend geprüft. Falls die kombinierte Vorzeichenüberlaufstelle nicht in beiden Ziffern den gleichen Wert hat, ist dies der Fall: Die Mantisse wird dann durch 10 dividiert und die Charakteristik um eins erhöht.

FLOATCOM
Die Gleitkommazahl in DAT1 wird komplementiert. Dies geschieht durch den Aufruf von COMBCD. Anschließend wird zur Marke NORM gesprungen und dort ein Überlauf korrigiert, der hier ebenfalls auftreten kann.

Abb. 3.3.6.1-3 zeigt einige Beispiele. Auf der Adresse 1000H wurde ein kleines Hauptprogramm geschrieben, das

```
>D1000 100F

1000  CD 03 01 CD 1E F0
>D1020 102F

1020  CD 06 01 CD 1E F0
>D200 20F

0200  50 00 00 12 55 66 00 00 50 00 00 56 88 77 22 00
>G1000
@1006
>D200 20F

0200  50 00 00 69 44 43 22 00 50 00 00 56 88 77 22 00
>G1020
>D200 20F

0200  50 00 99 30 55 56 78 00 50 00 00 56 88 77 22 00
>G1020
@1026
>D200 20F

0200  50 00 00 69 44 43 22 00 50 00 00 56 88 77 22 00


>D200 20F

0200  50 05 00 65 44 77 22 00 50 01 00 47 88 99 63 02
>G1000
@1006
>D200 20F

0200  50 05 00 65 45 25 10 99 50 05 00 00 00 47 88 99
>G1020
@1026
>D200 20F

0200  50 05 99 34 54 74 89 01 50 05 00 00 00 47 88 99


>D200 20F

0200  50 00 99 99 99 99 99 99 50 05 00 00 00 47 88 99
>G1020
@1026
>D200 20F

0200  50 00 00 00 00 00 00 01 50 05 00 00 00 47 88 99
```

Abb. 3.3.6.1-3 Rechenbeispiele mit der Gleitkommaarithmetik

```
>D200 20F

0200  50 00 99 00 00 00 00 00 50 05 00 00 00 47 88 99
>G1020
@1026
>D200 20F

0200  50 01 00 10 00 00 00 00 50 05 00 00 00 47 88 99
>G1020
@1026
>D200 20F

0200  50 01 99 90 00 00 00 00 50 05 00 00 00 47 88 99


>D200 20F

0200  50 01 00 45 66 77 88 03 50 01 00 45 66 77 88 45
>G1020
@1026
>D200 20F

0200  50 01 99 54 33 22 11 97 50 01 00 45 66 77 88 45
>G1000
@1006
>D200 20F

0200  50 01 00 00 00 00 00 42 50 01 00 45 66 77 88 45
```

Abb. 3.3.6.1-3

FLOATADD aufruft und auf 1020H ein Programm, das FLOATCOM aufruft.

Auf Adresse 200H befindet sich DAT1 und auf 208H DAT2. Einen kleinen Nachteil hat die hier beschriebene Arithmetik noch: Bei der Subtraktion von fast gleichen Zahlen tritt eine nicht normierte Zahl auf. Das letzte Beispiel zeigt eine solche Situation. Es wird dort .4566778845 E 0001 – .4566778803 E 0001 gebildet. Als Ergebnis erscheint: .0000000042 E 0001. Eine normierte Zahl würde sein: .4200000000 E –0007. Am Schluß von FLOATADD und FLOATCOM müßte also eine Normierung vorgenommen werden. Dieses Programm sieht an der höherwertigen Stelle nach, ob sich dort eine 0 befindet, oder im Falle einer negativen Zahl eine 9. Falls ja, so wird die Mantisse mit 10 multipliziert (durch eine Linksverschiebung) und die Charakteristik um eins verringert. Dabei muß eine Maximalzahl von 10 berücksichtigt werden, falls die Mantisse 0 ist.

Multiplikation und Division

Bei diesen Routinen müssen die Charakteristiken addiert bzw. subtrahiert werden und anschließend die Konstante 5000 subtrahiert bzw. addiert werden. Danach wird nach einer Betragsbildung der Mantissen die Multiplikation bzw. Division durchgeführt und anschließend die Vorzeichen wieder gebildet. Abschließend wird eine Normierung durchgeführt.

Beispiel:
5001 001234567890 x 5002 001234567890

1. Charakteristik
   addieren: 5001+5002→10003
2. 5000 subtra-
   hieren: 10003−5000 → 5003
3. Mantisse multi-
   plizieren: 1234567890 x 1234567890
   → 0152415787_...
4. Ergebnis vor Nor-
   mierung: 5003 000152415787
5. Normierung:
   5002 001524157870

Dabei ist die unterstrichene 0 eine fehlerhafte Stelle, die vermieden werden kann, wenn mit der doppelten Mantissenstellenzahl gerechnet wird, bis nach Ausführung der Normierung.

3.3.6.2 Floppy-Anschluß

Viel einfacher als ein Gleitkommapaket läßt sich eine Floppy in das Basic einbauen. Ist ein Betriebssystem (zum Beispiel CP/M oder FDOS) schon vorhanden, so ist dies besonders einfach: Dort gibt es Routinen zum Lesen und Schreiben von Dateien. Die Implementierung von SAVE und LOAD könnte dabei wie folgt aussehen:

SAVE dateiname
Der Parameter „dateiname" wird an das Betriebssystem übergeben und die Funktion ALLOCAT, OPEN ausgeführt, die die Datei anlegt und für Zugriffe freigibt. Dann wird über den Schreibbefehl der Inhalt des Basic-Programmspeichers auf die Floppy ausgegeben und am Schluß der Befehl CLOSE ausgeführt, ebenfalls eine Funktion, die ein Betriebssystem beinhaltet. Mit CLOSE wird der Rest der Datei auf die Floppy geschrieben und der Zugriff gesperrt.

LOAD dateiname
Hier wird nun nur ein OPEN durchgeführt und die Datei zum Lesen geöffnet. Dann wird mit READ die Datei in den Basic-Programmspeicher eingelesen; am Schluß werden noch die Pointer im Basic auf die neuen Werte eingestellt. CLOSE wird am Schluß ausgeführt, um die Funktion wieder zu sperren.

Außer LOAD und SAVE ist es wünschenswert, auch auf Dateiinhalte vom Basic aus zuzugreifen. Dazu müssen die Befehle OPEN und CLOSE vom Basic-Programm aus ausführbar sein. Ebenfalls muß eine Art ALLOCATE existieren, um neue Dateien anlegen zu können. Mit PRINT und INPUT kann auf diese Dateien zugegriffen werden, wobei sie dazu vom normalen PRINT und INPUT unterschieden werden müssen. Dazu wird meist ein Sondersymbol direkt hinter die Befehle gestellt. PRINT#1 würde z.B. auf den Kanal 1 ausgeben, wobei diese sogenannte Kanalnummer im OPEN-Befehl mit angegeben werden muß, um dem Kanal einen Dateinamen zuzuordnen.

Ist kein Betriebssystem vorhanden, so muß dies mit in den Basic-Interpreter eingebaut werden. Dabei kann leicht ein Umfang von 4...10 KByte allein für das Betriebssystem erreicht werden.

3.3.6.3 Grafik-Erweiterungen

Eine besondere Delikatesse ist Computergrafik. Gerade hier ist unser Basic-Interpreter ideal geeignet, da er eine sehr schnelle Arithmetik besitzt. Fehlende Routinen wie SIN können dabei leicht durch Tabellen ersetzt werden, auf die sehr schnell per Index zugegriffen werden kann. Hier einige Ideen für Grafikbefehle:

VECTOR (X1,Y1,X2,Y2)
Zeichnet einen Vektor von den Koordinatenpunkten X1,Y1 nach X2,Y2. Dies kann je nach vorhandenem Bildschirm oder Drucker mit unterschiedlicher Auflösung geschehen; zum Beispiel mit Pseudographik, die in einem Zeichengenerator vorhanden ist, oder mit hochauflösender Graphik, wenn

ein Bildschirm da ist, bei dem jeder einzelne Punkt ansteuerbar ist [5].

CLEAR
Löschen des gesamten Bildschirms.

LINE (M)
Setzen eines Linienmusters. Dies kann durchgehend sein, gepunktet, gestrichelt, löschend oder komplementierend und wirkt dann zum Beispiel auf den nächsten VECTOR-Befehl.

PLOT (X,Y)
Setzen eines einzelnen Punktes. Dies kann aber auch durch VECTOR erreicht werden (X1=X2, Y1=Y2).

CIRCLE (X,Y,r,phian,phiend)
Ein komplexerer Befehl, um einen Kreis zu zeichnen. Dazu werden Routinen für SIN und COS benötigt. Das kann aber leicht mit einer Tabelle, z.B. aus 256 Werten zu 8 Bit realisiert werden, wobei dort Sinuswerte von 0 bis 90 Grad gespeichert werden und diese Werte aus der Formel INT(SIN(phi)*256) gebildet werden. Dabei entstehen allerdings in der Nähe von 90 Grad Werte, die 256 betragen: Sie werden einfach durch 0 ersetzt und durch eine Abfrage in dem Hauptprogramm SIN wieder erzeugt. Ebenfalls wird im Hauptprogramm der Quadrant bestimmt, in dem der SIN-Wert gebildet werden soll, und daraus das Vorzeichen. Die Darstellung des SIN-Wertes kann nun wieder in INTEGER erfolgen, da die Werte nur von −256 bis +256 reichen können. Das Gleiche gilt für COS, wobei nur eine Verschiebung um 90 Grad erfolgt. Der Kreis wird nun in einer Schleife erzeugt, die wie folgt lauten könnte:

```
FOR I=phian TO phiend
DX=(COS'(I)*r)/256
DY=(SIN'(I)*r)/256
PLOT(X+DX,Y+DY)
NEXT I
```

Dabei ist SIN' und COS' die Spezialroutine mit dem Wertebereich −256 bis +256. Die Division durch 256 zur Normierung darf erst nach der Multiplikation mit r erfolgen, da sonst durch die INTEGER-Arithmetik ein Fehler entstehen würde.

```
Beispiel: SIN'(45) = 181 .  r=20
          DY=(181*20)/256
          DY= 14
aber      DY= 181*(20/256)
ergibt:   DY=0 !
```

Wird in DX und DY mit unterschiedlichen Werten für r multipliziert, so ergeben sich Ellipsen.

4 12-KByte-Basic für den Z80

Abb. 4-1 zeigt das Objektcode-Listing des „großen" Basic-Interpreters. Der Interpreter wurde ursprünglich von TDL (Technical Design Labs) entwickelt und ist sehr leistungsfähig. Da die Firma nicht mehr existiert, soll durch den Abdruck des Listings dem Leser die Möglichkeit gegeben werden, Zugang zu diesem Basic zu erhalten.

Abb. 4-1

```
0300 C3 C9 2F C3 73 04 C3 44 0A C3 03 F0 C3 06 F0 C3 ../.s..D........
0310 09 F0 C3 0C F0 C3 0F F0 C3 12 F0 C3 15 F0 C3 18 ................
0320 F0 C3 1B F0 C3 1E F0 C3 2D 0A C3 1E 12 A3 22 AD ........-.....".
0330 23 CE 22 06 03 F6 11 D4 16 2F 12 0E 2A 0E 2B EB #.".../..*.+.
0340 20 64 2A 4A 2B 50 2B D3 2B EA 2B 6A 1F 99 15 BA  d*J+P+.+.+j....
0350 13 2B 17 A8 15 B6 15 C6 15 F6 15 FF 15 2A 12 2E .+...........*..
0360 16 79 A7 24 79 8D 1F 7C 3B 21 7C BF 21 7F 17 2A .y.$y..|;!|.!..*
0370 50 BD 0F 46 BC 0F C3 09 6A 08 3B 0E 08 0B 55 0D P..F....j.;...U.
0380 65 10 A6 0D 27 0B C5 0A 1D 13 23 0C 6A 09 A8 0A e...'.....#.j...
0390 E6 0A 0D 0B C1 09 DD 16 06 0C FC 09 E3 16 36 12 ..............6.
03A0 71 1F 5C 0C 5C 0C 55 0D 72 0A 1D 13 C5 17 0D 0B q.\.\.U.r.......
03B0 0D 08 59 0C DB 1A D8 1A 84 1C 46 17 62 17 F9 09 ..Y.......F.b...
03C0 65 17 9E 1C 9B 1C 5C 0C 6D 04 59 1F 26 1F F9 1E e.....\.m.Y.&...
03D0 B6 1E 4F 0D B0 1D 9D 0A FE 17 B3 05 24 0D F2 1D ..O.........$...
03E0 6B 1D 6E 1D F2 1C F5 1C 41 18 EF 07 EC 07 B3 18 k.n.....A.......
03F0 A3 1A FE 1A E7 09 21 04 00 39 01 11 00 7E 23 FE ......!..9...~#.
0400 81 C0 7E 23 E5 66 6F 7A B3 EB 28 07 EB 7C 92 20 ..~#.foz..(..|.
0410 02 7D 93 E1 23 C8 09 18 E4 CD 3E 04 C5 E3 C1 B7 .}..#.....>.....
0420 E5 ED 52 C5 E3 C1 EB E3 03 ED B8 23 13 42 4B D1 ..R........#.BK.
0430 C9 E5 2A 62 01 06 00 09 09 CD 3E 04 E1 C9 D5 EB ..*b......>.....
0440 21 D8 FF 39 7C 92 20 02 7D 93 EB D1 D0 1E 07 18 !..9|. .}.......
0450 2D BE CA 47 09 18 16 3A 4F 01 B7 20 0A C1 21 84 -..G...:O.. ..!.
0460 2F CD BD 07 C3 F4 05 2A 4C 01 22 54 01 1E 02 01 /......*L."T....
0470 1E 0B 01 1E 16 01 1E 0A 01 1E 12 01 1E 01 CD DB ................
0480 05 CD F8 05 CD EE 0C 21 12 2E 1D 28 07 CB 7E 23 .......!...(..~#
0490 28 FB 18 F6 CD BD 07 2A 54 01 7C A5 3C C4 C1 24 (......*T.|.<..$
04A0 CD F8 05 32 AB 01 CD DB 0C 21 FF FF 22 54 01 3A ...2.....!.."T.:
04B0 AB 01 B7 28 25 2A 9A 01 FA D5 1D ED 5B 9C 01 19 ...(%*......[...
04C0 DA 6B 11 22 9A 01 E5 CD C9 24 CD 64 07 CD 07 07 .k.".....$.d....
04D0 CD 47 09 D1 B7 28 C9 37 18 0A CD 07 07 CD 47 09 .G...(.7......G.
04E0 3C 3D 28 C5 F5 FE 03 CA 90 09 D9 21 AB 01 34 35 <=(........!..45
04F0 D9 CC 49 0A FE 20 20 01 23 D5 CD 09 06 47 D1 F1 ..I..  .#....G..
0500 D2 14 09 D5 C5 23 7E B7 28 0C FE 20 28 F7 FE 09 .....#~.(.. (...
0510 28 F3 FE 0A 28 EF F5 CD 92 05 C5 30 14 EB 2A 5E (...(......0..*^
0520 01 1A 02 03 13 7C 92 20 02 7D 93 20 F4 ED 43 5E .....|. .}. ..C^
0530 01 D1 F1 28 21 ED 4B 5E 01 E1 09 E5 CD 19 04 E1 ...(!.K^........
0540 22 5E 01 EB 74 23 23 D1 73 23 72 23 11 B1 01 1A "^..t##.s#r#....
0550 77 23 13 B7 20 F9 CD C2 05 23 EB 21 A9 04 E5 62 w#.. ....#.!...b
0560 6B 7E 23 B6 C8 23 23 23 AF BE 23 20 FC EB 73 23 k~#..###..# ..s#
0570 72 18 EC 11 00 00 D5 28 0C D1 CD 85 09 D5 28 0E r......(......(.
0580 3E A6 CD 51 04 11 FF FF C4 85 09 C2 6D 04 EB D1 >..Q........m...
```

```
0590 E3 E5 2A 5C 01 44 4D 7E 23 B6 2B C8 23 23 7E 23  ..*\.DM~#.+.##~#
05A0 66 6F ED 52 60 69 7E 23 66 6F 3F C8 3F D0 D9 19  fo.R`i~#fo?.?...
05B0 D9 18 E2 C0 2A 5C 01 AF 32 A3 01 77 23 77 23 22  ....*\..2..w#w#"
05C0 5E 01 2A 5C 01 2B 22 50 01 2A 04 01 22 48 01 CD  ^.*\.+"P.*.."H..
05D0 7B 09 2A 5E 01 22 60 01 22 62 01 C1 ED 7B 5A 01  {.*^."`."b...{Z.
05E0 21 08 01 22 06 01 AF 67 6F 22 58 01 32 4E 01 32  !.."...go"X.2N.2
05F0 A7 01 E5 C5 2A 50 01 C9 AF 32 00 01 FD 21 0F 03  ....*P...2...!..
0600 FD 22 8E 01 FD 21 90 01 C9 AF 32 03 01 0E 05 11  ."...!....2.....
0610 B1 01 7E 47 B7 CA AD 06 FE 21 38 4F FE 22 28 7B  ..~G.....!8O."({
0620 3A 03 01 B7 47 7E 20 43 FE 30 38 04 FE 3A 38 3B  :...G~ C.08..:8;
0630 D5 11 63 2C E5 18 02 23 13 1A B7 28 25 FE 20 20  ..c,...#...(%.
0640 09 13 7E FE 20 20 03 23 18 F8 EB 1A FE 60 38 02  ..~.  .#.....`8.
0650 E6 5F AE E6 7F EB 20 49 1A 17 30 DB F1 78 F6 80  ._.... I..0..x..
0660 18 08 E1 7E FE 60 38 02 E6 5F D1 EB FE C9 36 3A  ...~.`8.._....6:
0670 20 02 0C 23 EB 23 12 13 0C D6 3A 28 0A FE 9B 06   ..#.#....:(....
0680 3A 28 11 FE 49 20 03 32 03 01 D6 54 28 05 D6 0E  :(..I .2...T(...
0690 C2 12 06 47 7E B7 28 15 B8 28 D0 23 12 0C 13 18  ...G~.(..(.#....
06A0 F3 E1 E5 04 EB CB 7E 23 28 FB EB 18 8C 21 B0 01  ......~#(....!..
06B0 12 13 12 13 12 C9 78 3D FD B6 00 28 4A CD 70 07  ......x=...(J.p.
06C0 10 08 CD 70 07 CD 70 07 18 3A 2B 7E CD 66 07 CD  ...p..p..:+~.f..
06D0 C9 07 FE 7F 28 EA F5 CD 70 07 F1 18 36 3E 3F CD  ....(...p...6>?.
06E0 72 07 3E 3F CD 72 07 CD 64 07 CD 07 07 23 7E 2B  r.>?.r..d....#~+
06F0 FE 03 C9 CD FC 05 F5 3E 5E CD 72 07 F1 F6 40 18  .......>^.r...@.
0700 71 CD F6 06 CD 0E 0D 21 B1 01 06 01 78 32 A7 01  q......!....x2..
0710 CD C9 07 FE 07 28 3C FE 0D CA E9 0C FE 03 CA E1  .....(<.........
0720 0C FE 15 28 DC FE 7F 28 8D FE 12 20 14 CD F6 06  ...(...(... ....
0730 CD 0E 0D 21 B1 01 48 0D 28 D6 7E 23 CD 66 07 18  ...!..H.(.~#.f..
0740 F6 FE 09 28 0E FE 0A 20 06 05 28 BB 04 18 04 FE  ...(... ..(.....
0750 20 38 BD 4F 78 FE FD 3E 07 30 04 79 77 23 04 CD   8.Ox..>.0.yw#..
0760 66 07 18 AC 3E 20 FE 0A 20 08 CD EE 0C 3E 0A C9  f...> .. .....>..
0770 3E 5C C5 4F 3A 00 01 B7 20 3F 79 FE 09 20 27 FD  >\.O:... ?y.. '.
0780 7E 01 E6 F8 3D FD BE 00 38 17 FD 7E 00 E6 07 2F  ~...=...8..~.../
0790 C6 09 47 0E 20 CD 8D 01 FD 34 00 10 F8 0E 09 18  ..G. ....4.......
07A0 18 CD EE 0C 18 F7 FE 20 38 0C FD 7E 00 FD BE 01  ....... 8..~....
07B0 CC EE 0C FD 34 00 CD 8D 01 79 C1 B7 C9 7E CB BF  ....4....y...~..
07C0 CD 66 07 CB 7E C0 23 18 F4 CD 09 03 E6 7F FE 18  .f..~.#.........
07D0 28 0F FE 0F C0 CD F3 06 3A 00 01 2F 32 00 01 AF  (.......:../2...
07E0 C9 CD F3 06 CD EE 0C CD 24 03 AF C9 CD A6 18 C1  .........$.......
07F0 CD 73 05 C5 CD EE 0C E1 D1 4E 23 46 23 78 B1 28  .s.......N#F#x.(
0800 33 CD 93 09 C5 4E 23 46 23 C5 E3 EB 7C 92 20 02  3....N#F#...|. .
0810 7D 93 C1 38 1E E3 E5 C5 EB CD C9 24 E1 CD 64 07  }..8........$..d.
0820 CD 37 08 21 B1 01 01 F4 07 C5 2B 06 00 CD F0 13  .7.!......+.....
0830 C3 34 14 C1 C3 A0 04 01 B0 01 3E E1 7E 03 B7 23  .4........>.~..#
0840 02 C8 F2 3C 08 FE C9 20 01 0B D6 80 E5 21 63 2C  ...<... ......!c,
0850 28 08 CB 7E 23 28 FB 3D 20 F8 7E FE 20 28 08 B7  (..~#(.= .~. (..
0860 CB BF 02 FA 3B 08 03 23 18 F0 3E 29 32 4E 01 CD  ....;..#..>)2N..
0870 27 0B E3 CD F6 03 D1 20 02 09 F9 EB 0E 0A CD 31  '....... .......1
0880 04 E5 CD 0B 0B E3 E5 2A 54 01 E3 3E 9F CD 51 04  .......*T..>..Q.
0890 CD 88 0E CD 85 0E E5 CD 02 23 E1 C5 DD E5 D5 01  .........#......
08A0 00 81 DD 21 00 00 51 59 7E FE A4 3E 01 20 0E CD  ...!..QY~..>. ..
08B0 47 09 CD 85 0E E5 CD 02 23 CD BF 22 E1 C5 DD E5  G.......#.."....
08C0 D5 F5 33 E5 2A 50 01 E3 06 81 C5 33 CD 18 03 3C  ..3.*P.....3...<
08D0 CC 98 09 22 50 01 7E FE 3A 28 39 B7 C2 6D 04 23  ..."P.~.:(9..m.#
08E0 7E 23 B6 23 CA C8 09 5E 23 56 ED 53 54 01 3A A3  ~#.#...^#V.ST.:.
08F0 01 B7 28 20 F5 FC A6 18 D5 E5 3E 3C CD 72 07 EB  ..( ......><.r..
0900 CD C9 24 3E 3E CD 72 07 E1 D1 F1 87 30 06 CD FC  ..$>>.r.....0...
0910 05 87 38 E0 CD 47 09 11 CC 08 D5 C8 D6 80 DA 2B  ..8..G.........+
0 0B FE 46 CA 8B 0B FE 1D 38 0F D6 49 DA 6D 04 FE  ..F.....8..I.m..
0930 23 D2 6D 04 11 B0 03 18 03 11 76 03 07 4F 06 00  #.m.......v..O..
0940 EB 09 4E 23 46 C5 EB 23 7E FE D5 20 08 23 7E FE  ..N#F..#~.. .#~.
0950 3A C8 B7 20 F8 FE 3A D0 FE 20 28 EB FE 09 28 E7  :.. ..:.. (...(.
0960 FE 0A 28 E3 FE 30 3F 3C 3D C9 28 0F CD 85 09 E5  ..(..0?<=.(.....
```

Abb. 4-1 Listing des 12-K-Basic-Interpreters

```
0970 CD 92 05 D1 D2 E1 0A 60 69 18 04 EB 2A 5C 01 2B ........`i...*\.+
0980 22 64 01 EB C9 2B CD 47 09 CD 49 0A 18 BA C1 C1 "d...+.G..I.....
0990 B7 18 34 CD 18 03 3C C0 CD C9 07 FE 14 20 0C E5 ..4...<...... ..
09A0 2A 54 01 7C A5 3C C4 9D 1A E1 C9 FE 13 20 0B CD *T.!.<...... ..
09B0 C9 07 FE 03 28 04 FE 11 20 F5 FE 03 C0 CD F3 06 ....(... .......
09C0 3E C0 F6 C0 22 50 01 C1 F5 2A 54 01 7D A4 3C 28 >..."P...*T.}.<(
09D0 09 22 56 01 2A 50 01 22 58 01 CD F8 05 F1 21 AF ."V.*P."X.....!.
09E0 2F C2 94 04 C3 A0 04 1E 11 2A 58 01 7C B5 28 56 /........*X.!.(V
09F0 EB 2A 56 01 22 54 01 EB C9 CD 9F 18 CD 1B 17 FE .*V."T..........
0A00 32 30 41 FD 77 03 3E 2C BE C0 CD 47 09 CD 1B 17 20A.w.>,...G....
0A10 FD 77 04 C9 7E FE 41 D8 FE 5B 3F C9 CD 88 0E 18 .w..~.A..[?.....
0A20 0C CD 47 09 CD 85 0E CD BF 22 FA 44 0A 3A 6B 01 ..G......".D.:k.
0A30 FE 91 DA 75 23 01 80 91 DD 21 00 00 11 00 00 CD ...u#....!......
0A40 36 23 51 C8 1E 05 C3 7E 04 2B 11 00 00 23 7E B7 6#Q....~.+...#~.
0A50 C8 FE 30 3F D0 FE 3A D0 E5 21 98 19 B7 ED 52 DA ..0?..:..!....R.
0A60 6D 04 62 6B 19 29 19 29 D6 30 5F 16 00 19 EB E1 m.bk.).).0_.....
0A70 18 DB 28 26 CD 24 0A CD 48 09 C0 E5 2A 04 01 ED ..(&.$..H...*...
0A80 52 EB DA 6D 04 2A 5E 01 01 28 00 09 7C 92 20 02 R..m.*^..(..!. .
0A90 7D 93 D2 4D 04 EB 22 5A 01 E1 C3 C6 05 CA C2 05 }..M.."Z........
0AA0 CD C6 05 01 CC 08 18 1C 0E 03 CD 31 04 C1 E5 ED ...........1....
0AB0 5B 54 01 D5 7A B3 3C 3A A7 01 57 20 04 AF 32 A7 [T..z.<:..W ..2.
0AC0 01 1E 8C D5 C5 CD 49 0A CD 0D 0B E5 2A 54 01 7C ......I.....*T.!
0AD0 92 20 02 7D 93 E1 23 DC 95 05 D4 92 05 60 69 2B . .}..#......`i+
0AE0 D8 1E 08 C3 7E 04 C0 16 FF CD F6 03 56 23 F9 FE ....~.......V#..
0AF0 8C 1E 03 20 EE E1 22 54 01 7C A5 3C 21 A7 01 7E ... .."T.!.<!..~
0B00 72 20 04 B7 C2 A5 13 21 CC 08 E3 01 3A 0E 00 06 r .....!....:...
0B10 00 79 48 47 7E B7 C8 B8 C8 23 FE 22 28 F3 D6 8A .yHG~....#."(...
0B20 20 F2 B8 8A 57 18 ED FE C6 28 60 CD 6A 10 3E AD  ...W....(`.j.>.
0B30 CD 51 04 D5 3A 02 01 F5 CD 9A 0E F1 E3 22 50 01 .Q..:........"P.
0B40 1F CD 8A 0E 20 07 E5 CD 16 23 D1 E1 C9 E5 2A 66 .... ....#....*f
0B50 01 E5 23 23 5E 23 56 21 B0 02 7C 92 20 02 7D 93 ..##^#V!..!. .}.
0B60 30 18 2A 5A 01 7C 92 20 02 7D 93 D1 30 14 2A 5E 0.*Z.!. .}..0.*^
0B70 01 7C 92 20 02 7D 93 30 09 3E D1 CD 83 15 EB CD .!. .}.0.>......
0B80 CA 13 CD 83 15 E1 CD 19 23 E1 C9 CD 47 09 3E 28 ........#...G.>(
0B90 CD 51 04 CD 6A 10 CD 89 0E 3E 2C CD 51 04 D5 CD .Q..j....>,.Q...
0BA0 1B 17 B7 CA 44 0A F5 1E FF 7E FE 29 28 08 3E 2C ....D....~.)(.>,
0BB0 CD 51 04 CD 1B 17 3E 29 CD 51 04 3E AD CD 51 04 .Q....>).Q.>..Q.
0BC0 F1 E3 3D BE 06 00 30 08 4F 7E 91 BB 47 38 01 43 ..=...0.O~..G8.C
0BD0 C5 23 23 46 23 66 68 06 00 09 C1 E3 C5 CD 95 0E .##F#fh.........
0BE0 C1 D1 E5 2A 66 01 78 96 F5 78 38 01 7E 23 23 4E ...*f.x..x8.~##N
0BF0 23 46 CD 59 15 F1 38 09 28 07 EB 36 20 23 3D 20 #F.Y..8.(..6 #= 
0C00 FA CD 63 15 E1 C9 CD 1B 17 7E 47 FE 8C 28 06 3E ..c......~G..(.>
0C10 88 CD 51 04 2B 4B 0D 78 CA 1C 09 CD 86 09 FE 2C ..Q.+K.x.......,
0C20 C0 18 F3 CD 9A 0E 7E FE 2C CC 47 09 FE 88 28 06 ......~.,.G...(.
0C30 3E A2 CD 51 04 2B E5 CD BF 22 E1 28 09 CD 47 09 >..Q.+...".(..G.
0C40 DA C5 0A C3 1B 09 16 01 CD 0B 0B B7 C8 CD 47 09 ..............G.
0C50 FE C9 20 F4 15 20 F1 18 E4 CD 9F 18 CA EE 0C FE .. .. ..........
0C60 9D CA 21 28 FE 9E 28 4E FE A1 28 49 E5 FE 2C 28 ..!(..(N..(I..,(
0C70 32 FE 3B 28 5F C1 CD 9A 0E E5 3A 02 01 B7 20 1A 2.;(_.....:... .
0C80 CD DD 24 CD ED 13 2A 66 01 FD 7E 00 86 FD BE 01 ..$...*f..~.....
0C90 D4 EE 0C CD 34 14 CD 64 07 AF C4 34 14 E1 CD 48 ....4..d...4...H
0CA0 09 18 B9 FD 7E 00 FD BE 02 D4 EE 0C 30 26 D6 0E ....~.......0&..
0CB0 30 FC 2F 18 18 37 F5 CD 1B 17 3E 29 CD 51 04 2B 0./..7....>).Q.+
0CC0 F1 E5 3E FF 38 04 FD 7E 00 2F 83 30 07 3C 47 CD ..>.8..~./.0.<G.
0CD0 64 07 10 FB E1 CD 47 09 C8 18 81 21 9B 2F C3 A4 d.....G....!./..
0CE0 01 21 B1 01 77 23 CD F6 06 36 00 21 B0 01 3E 0D .!..w#...6.!..>.
0CF0 FD 77 00 CD 72 07 3E 0A CD 72 07 FD 7E 03 3C 3D .w..r.>..r..~.<=
0D00 FD 77 00 C8 F5 FD 7E 04 CD 72 07 F1 18 F1 CD EE .w....~..r......
0D10 0C 3A AB 01 B7 C8 E5 C5 2A 9A 01 CD C9 24 CD 64 .:......*....$.d
0D20 07 C1 E1 C9 D1 11 0A 00 D5 DC 85 09 EB E3 EB FE ................
0D30 2C 20 06 CD 47 09 DC 85 09 B7 C2 6D 04 ED 53 9C , ..G......m..S.
0D40 01 7A B3 CA 6D 04 3E 01 32 AB 01 E1 C3 C3 04 3E .z..m.>.2......>
```

Abb. 4-1

```
0D50 84 CD 51 04 F6 AF F5 7E FE 22 3E 00 32 00 01 20 ..Q....~.">.2..
0D60 0D CD EE 13 3E 3B CD 51 04 E5 CD 34 14 E1 E3 E5 ....>;.Q...4....
0D70 CD 9C 13 CD E2 06 CA 8E 09 F1 28 2F E3 CD 6A 10 ..........(/..j.
0D80 CD 89 0E E3 D5 06 00 CD F0 13 EB 21 93 0D E3 D5 ...........!....
0D90 C3 4D 0B E1 CD 48 09 C8 3E 2C CD 51 04 E5 CD DD .M...H..>,.Q....
0DA0 06 CA 8F 09 18 D6 E5 2A 64 01 F6 AF 32 4F 01 E3 .......*d...2O..
0DB0 18 05 3E 2C CD 51 04 CD 6A 10 E3 D5 7E FE 2C 28 ..>,.Q..j...~.,(
0DC0 0C 3A 4F 01 B7 20 54 CD DD 06 CA 8E 09 3A 02 01 .:O.. T......:..
0DD0 B7 28 1A CD 47 09 57 47 FE 22 28 05 16 3A 06 2C .(..G.WG."(..:.,
0DE0 2B CD F1 13 EB 21 F8 0D E3 D5 C3 4D 0B CD 47 09 +....!.....M..G.
0DF0 CD DE 23 E3 CD 16 23 E1 CD 48 09 28 05 FE 2C C2 ..#...#..H.(..,.
0E00 57 04 E3 CD 48 09 20 AA D1 3A 4F 01 B7 EB C2 80 W...H. ..:O.....
0E10 09 B6 21 A3 2F D5 C4 BD 07 E1 C9 CD 0B 0B B7 20 ..!./...........
0E20 10 23 7E 23 B6 1E 04 28 69 23 5E 23 56 ED 53 4C .#~#...(i#^#V.SL
0E30 01 CD 47 09 FE 83 20 E3 C3 CD 0D 11 00 00 C4 6A ..G... .........j
0E40 10 22 50 01 CD F6 03 C2 7C 04 F9 D5 7E 23 F5 D5 ."P.....|...~#..
0E50 CD EC 22 E3 E5 CD 83 1F E1 CD 16 23 E1 CD 05 23 .."........#...#
0E60 E5 CD 36 23 E1 C1 90 CD 6C 23 28 09 EB 22 54 01 ..6#....l#(.."T.
0E70 69 60 C3 C8 08 F9 2A 50 01 7E FE 2C C2 CC 08 CD i`....*P.~.,....
0E80 47 09 CD 3E 0E CD 9A 0E F6 37 3A 02 01 8F B7 E8 G..>.....7:.....
0E90 1E 0D C3 7E 04 CD 9A 0E 18 EF 2B 16 00 D5 0E 01 ...~......+.....
0EA0 CD 31 04 CD 14 0F 22 52 01 2A 52 01 C1 78 FE 78 .1...."R.*R..x.x
0EB0 D4 88 0E 7E 16 00 D6 AC 38 15 FE 03 30 11 FE 01 ...~....8...0...
0EC0 17 AA BA 57 DA 6D 04 22 4A 01 CD 47 09 18 E7 7A ...W.m."J..G...z
0ED0 B7 C2 E2 0F 7E 22 4A 01 D6 A5 D8 FE 07 D0 5F 3A ....~"J......._:
0EE0 02 01 3D B3 7B CA 12 15 07 83 5F 21 61 03 19 78 ..=.{....._!a..x
0EF0 56 BA D0 23 CD 88 0E C5 01 A9 0E C5 ED 4B 66 01 V..#.........Kf.
0F00 C5 ED 4B 68 01 C5 ED 4B 6A 01 C5 4E 23 46 C5 2A ..Kh...Kj..N#F.*
0F10 4A 01 18 89 AF 32 02 01 CD 47 09 DA E3 23 CD 14 J....2...G...#..
0F20 0A 30 44 FE A5 28 ED FE 2E CA E3 23 FE A6 28 26 .0D..(.....#..(&
0F30 FE 22 CA EE 13 FE A3 CA 45 10 FE A0 CA 6C 12 FE ."......E....l..
0F40 26 CA 71 24 D6 AF 30 30 3E 28 CD 51 04 CD 9A 0E &.q$..00>(.Q....
0F50 3E 29 CD 51 04 C9 16 7D CD 9D 0E 2A 52 01 E5 CD >).Q...}...*R...
0F60 D2 22 CD 88 0E E1 C9 CD 6A 10 E5 EB 22 66 01 3A ."......j..."f.:
0F70 02 01 B7 CC EC 22 E1 C9 06 00 07 4F C5 CD 47 09 .....".....O..G.
0F80 79 FE 29 38 26 FE 30 28 22 FE 32 28 1B D2 44 0A y.)8&.0(".2(..D.
0F90 3E 28 CD 51 04 CD 95 0E 3E 2C CD 51 04 EB 2A 66 >(.Q....>,.Q..*f
0FA0 01 E3 E5 EB CD 1B 17 EB E3 18 08 CD 48 0F E3 11 ............H...
0FB0 62 0F D5 01 2D 03 09 4E 23 66 69 E9 F6 AF F5 CD b...-..N#fi.....
0FC0 1C 0A F1 EB C1 DD E1 E3 EB CD F5 22 F5 CD 2D 0A ..........."..-.
0FD0 F1 C1 79 21 1D 12 20 05 A3 4F 78 A2 E9 B3 4F 78 ..y!.. ..Ox...Ox
0FE0 B2 E9 21 F4 0F 3A 02 01 1F 7A 17 5F 16 64 78 BA ..!..:...z._.dx.
0FF0 D0 C3 F7 0E F6 0F 79 B7 1F C1 DD E1 D1 F5 CD 8A ......y.........
1000 0E 21 3B 10 E5 CA 36 23 AF 32 02 01 D5 CD 63 15 .!;...6#.2....c.
1010 D1 4E 23 46 23 C5 4E 23 46 C5 CD 67 15 CD 6C 23 .N#F#.N#F..g..l#
1020 E1 E3 55 E1 7B B2 C8 7A D6 01 D8 AF BB 3C D0 15 ..U.{..z.....<..
1030 1D 0A BE 23 03 28 ED 3F C3 CA 22 3C 8F C1 A0 C6 ...#.(.?.."<....
1040 FF 9F C3 A6 22 16 5A CD 9D 0E CD 88 0E CD 2D 0A ....".Z.......-.
1050 7B 2F 4F 7A 2F CD 1D 12 C1 C3 A9 0E CD 48 09 C8 {/Oz/........H..
1060 3E 2C CD 51 04 01 5C 10 C5 F6 AF 32 01 01 46 CD >,.Q..\....2..F.
1070 14 0A DA 6D 04 AF 4F 32 02 01 CD 47 09 38 05 CD ...m..O2...G.8..
1080 14 0A 38 0B 4F CD 47 09 38 FB CD 14 0A 30 F6 D6 ..8.O.G.8....0..
1090 24 20 09 3C 32 02 01 CB F9 CD 47 09 3A 4E 01 3D $ .<2.....G.:N.=
10A0 CA 3C 11 86 D6 27 28 6C AF 32 4E 01 E5 2A 60 01 .<...'(l.2N..*`.
10B0 EB 2A 5E 01 7C 92 20 02 7D 93 28 12 79 96 23 20 .*^.|. .}.(.y.#
10C0 02 78 96 23 28 40 23 23 23 23 23 23 18 E6 E1 E3 .x.#(@######....
10D0 D5 11 6A 0F 7C 92 20 02 7D 93 D1 28 2C E3 E5 C5 ..j.|. .}..(,...
10E0 01 08 00 2A 62 01 E5 09 C1 E5 CD 19 04 E1 22 62 ...*b.........."b
10F0 01 60 69 22 60 01 2B 36 00 7C 92 20 02 7D 93 20 .`i"`.+6.|. .}.
1100 F5 D1 73 23 72 23 EB E1 C9 32 6B 01 21 FF 02 22 ..s#r#...2k.!.."
1110 66 01 E1 C9 E5 2A 01 01 E3 57 D5 C5 CD 21 0A C1 f....*...W...!..
1120 F1 EB E3 E5 EB 3C 57 7E FE 2C 28 EE 3E 29 CD 51 .....<W~.,(.>).Q
```

Abb. 4-1

```
1130 04 22 52 01 E1 22 01 01 1E 00 D5 11 E5 F5 2A 60 ."R.."........*`
1140 01 3E 19 ED 5B 62 01 7C 92 20 02 7D 93 28 21 7E .>..[b.|. .}.(!~
1150 B9 23 20 02 7E B8 23 5E 23 56 23 20 E5 3A 01 01 .# .~.#^#V# .:..
1160 B7 C2 76 04 F1 CA 01 1F 96 28 61 1E 09 C3 7E 04 ..v......(a...~.
1170 11 06 00 F1 CA 19 1F 71 23 70 23 4F CD 31 04 23 .......q#p#O.1.#
1180 23 22 4A 01 71 23 3A 01 01 17 79 01 0B 00 30 02 #"J.q#:...y...0.
1190 C1 03 71 23 70 23 F5 E5 CD C6 23 EB E1 F1 3D 20 ..q#p#....#...=
11A0 EA F5 42 4B EB 19 38 C3 CD 3E 04 22 62 01 2B 36 ..BK..8..>."b.+6
11B0 00 7C 92 20 02 7D 93 20 F5 03 57 2A 4A 01 5E EB .|. .}. ..W*J.^.
11C0 29 09 EB 2B 2B 73 23 72 23 F1 38 26 47 4F 7E 23 )..++s#r#.8&GO~#
11D0 16 E1 5E 23 56 23 E3 F5 7C 92 20 02 7D 93 30 8B ..^#V#..|. .}.0.
11E0 E5 CD C6 23 D1 19 F1 3D 44 4D 20 E5 29 09 29 C1 ...#...=DM .).).
11F0 09 EB 2A 52 01 C9 2A 62 01 EB 21 00 00 39 3A 02 ..*R..*b..!..9:.
1200 01 B7 28 0D CD 63 15 CD 73 14 2A 5A 01 EB 2A 48 ..(..c..s.*Z..*H
1210 01 AF 32 02 01 ED 52 EB AF 06 98 18 0A 41 50 1E ..2...R......AP.
1220 00 21 02 01 73 06 90 C3 AB 22 3A 95 01 18 03 3A .!..s....":....:
1230 90 01 47 AF 18 E8 CD AA 13 CD 9C 13 EB 73 23 72 ..G..........s#r
1240 EB 2B CD 47 09 28 07 FE AD 20 F7 C3 0B 0B B7 20 .+.G.(... ......
1250 0F 23 7E 23 B6 CA 6D 04 23 5E 23 56 ED 53 54 01 .#~#..m.#^#V.ST.
1260 CD 47 09 28 E9 FE 89 28 E2 C3 60 12 CD AA 13 3A .G.(...(..`....:
1270 02 01 B7 F5 22 52 01 EB 7E 23 66 6F B4 CA 79 04 ...."R..~#fo..y.
1280 7E FE 28 20 71 CD 47 09 22 4A 01 18 05 3E 2C CD ~.( q.G."J...>,.
1290 51 04 0E 05 CD 31 04 3E 29 32 4E 01 CD 6A 10 EB Q....1.>)2N..j..
12A0 3A 02 01 B7 37 20 10 4E 23 46 C5 23 4E 23 46 C5 :...7 .N#F.#N#F.
12B0 23 4E 23 46 C5 18 08 F5 D5 EB CD 10 14 D1 F1 E5 #N#F............
12C0 F5 EB 7E FE 29 20 C6 2A 52 01 3E 28 CD 51 04 E5 ..~.) .*R.>(.Q..
12D0 2A 4A 01 CD 6A 10 E3 CD 33 08 7E FE 29 28 0D 3E *J..j...3.~.)(.>
12E0 2C CD 51 04 E3 3E 2C CD 51 04 18 E7 CD 47 09 E3 ,.Q..>,.Q....G..
12F0 3E 29 CD 51 04 3E D5 CD 48 09 28 10 3E AD CD 51 >).Q.>..H.(.>..Q
1300 04 CD 9A 0E CD 48 09 C2 6D 04 18 3A 0E 02 CD 31 .....H..m..:...1
1310 04 ED 5B 54 01 D5 16 A0 D5 33 C3 CC 08 20 0C CD ..[T.....3... ..
1320 0E 20 32 44 01 2F 32 02 01 18 09 CD 9A 0E CD 48 . 2D./2........H
1330 09 C2 6D 04 16 FF CD F6 03 F9 FE A0 1E 17 C2 7E ..m............~
1340 04 D1 ED 53 54 01 3A 02 01 3C 28 03 3D 20 2E D1 ...ST.:..<(.= ..
1350 F1 30 13 20 37 E1 C1 70 2B 71 2B C1 70 2B 71 2B .0. 7..p+q+.p+q+
1360 C1 70 2B 71 18 EA F5 D5 21 02 01 CB 7E 28 01 77 .p+q....!...~(.w
1370 B7 11 44 01 C4 10 14 E1 F1 1F C3 8A 0E ED 5B 66 ..D...........[f
1380 01 CD 83 15 21 44 01 CD 19 23 18 C3 CD 83 15 7E ....!D...#.....~
1390 22 06 01 E1 77 23 23 71 23 70 18 B4 E5 2A 54 01 "...w##q#p...*T.
13A0 23 7C B5 E1 C0 1E 0C C3 7E 04 3E A0 CD 51 04 F6 #|......~.>..Q..
13B0 80 47 3E 29 32 4E 01 C3 6F 10 CD 88 0E CD DD 24 .G>)2N..o......$
13C0 CD ED 13 CD 63 15 01 C2 15 C5 7E 23 23 E5 CD 49 ....c.....~##..I
13D0 14 E1 4E 23 46 CD E1 13 E5 CD 59 15 D1 C9 CD 49 ..N#F.....Y....I
13E0 14 21 44 01 E5 77 23 23 73 23 72 E1 C9 2B 06 22 .!D..w##s#r..+."
13F0 50 E5 0E FF 23 7E 0C B7 28 06 BA 28 03 B8 20 F4 P...#~..(..(.. .
1400 FE 22 CC 47 09 E3 23 EB 79 CD E1 13 11 44 01 3E .".G..#.y....D.>
1410 D5 2A 06 01 22 66 01 3E 01 32 02 01 CD 19 23 7C .*.."f.>.2....#|
1420 92 20 02 7D 93 22 06 01 E1 7E C0 1E 10 C3 7E 04 . .}."...~.....~.
1430 23 CD ED 13 CD 63 15 CD 6C 23 1C 1D C8 0A CD 66 #....c..l#.....f
1440 07 FE 0D CC FB 0C 03 18 F2 B7 0E F1 F5 2A 5A 01 .............*Z.
1450 EB 2A 48 01 4F AF 47 ED 42 7C 92 20 02 7D 93 38 .*H.O.G.B|. .}.8
1460 07 22 48 01 23 EB F1 C9 F1 1E 0E 28 C0 BF F5 01 ."H.#......(....
1470 4B 14 C5 2A 04 01 22 48 01 AF 08 ED 5B 5A 01 D9 K..*.."H....[Z..
1480 21 08 01 ED 5B 06 01 7C 92 20 02 7D 93 01 83 14 !...[..|. .}....
1490 20 3F 2A 5E 01 ED 5B 60 01 7C 92 20 02 7D 93 28  ?*^..[`.|. .}.(
14A0 0A CB 7E 23 23 CD D2 14 18 EB C1 EB 2A 62 01 ED ..~##.......*b..
14B0 52 EB 28 44 CD 6C 23 CB 7B E5 09 28 ED E3 4E 06 R.(D.l#.{..(..N.
14C0 00 09 09 23 D1 7C 92 20 02 7D 93 28 DE D5 01 C4 ...#.|. .}.(....
14D0 14 C5 7E 23 23 5E 23 56 23 23 23 C8 B7 C8 D5 D9 ..~##^#V###.....
14E0 C1 2A 48 01 ED 42 D9 D8 D9 62 6B ED 42 D9 D0 D9 .*H..B...bk.B...
14F0 50 59 D9 08 E5 DD E1 C9 08 D0 D9 2A 48 01 EB 4F PY.........*H..O
1500 06 00 09 2B ED B8 62 6B 13 DD 72 FD DD 73 FC C3 ...+..bk..r..s..
```

Abb. 4-1

```
1510 76 14 C5 E5 2A 66 01 E3 CD 14 0F E3 CD 89 0E 7E v...*f.........~
1520 E5 2A 66 01 E5 86 1E 0F DA 7E 04 CD DE 13 D1 CD .*f......~......
1530 67 15 E3 CD 66 15 E5 2A 46 01 EB CD 49 15 CD 49 g...f..*F...I..I
1540 15 21 AC 0E E3 E5 C3 0C 14 E1 E3 4E 23 46 23 7E .!.........N#F#~
1550 23 66 6F 78 B1 C8 ED B0 C9 60 69 4F 06 00 18 F3 #fox.....`iO....
1560 CD 89 0E 2A 66 01 EB CD 83 15 EB C0 D5 50 59 1B ...*f........PY.
1570 4E 2A 48 01 7C 92 20 02 7D 93 20 05 47 09 22 48 N*H.!. .}. .G."H
1580 01 E1 C9 2A 06 01 2B 2B 2B 46 2B 4E 2B 2B 7C 92 ...*..+++F+N++!.
1590 20 02 7D 93 C0 22 06 01 C9 01 32 12 C5 CD 60 15  .}.."....2...`.
15A0 AF 57 32 02 01 7E B7 C9 CD 9D 15 28 58 23 23 5E .W2..~.....(X##^
15B0 23 56 1A C3 32 12 3E 01 CD DE 13 CD 1E 17 2A 46 #V..2.>.......*F
15C0 01 73 C1 C3 0C 14 CD FF 16 AF E3 4F 3E E5 E5 7E .s.........O>..~
15D0 B8 38 02 78 11 0E 00 C5 CD 49 14 C1 E1 E5 23 23 .8.x.....I.....##
15E0 46 23 66 68 06 00 09 44 4D CD E1 13 CD 59 15 D1 F#fh...DM....Y..
15F0 CD 67 15 C3 0C 14 CD FF 16 D1 D5 1A 90 18 CB EB .g..............
1600 7E D1 43 04 05 CA 44 0A C5 1E FF FE 29 28 08 3E ~.C...D.....)(.>
1610 2C CD 51 04 CD 1B 17 3E 29 CD 51 04 F1 E3 3D BE ,.Q....>).Q...=.
1620 06 00 30 AA 4F 7E 91 BB 47 38 A3 43 18 A0 E1 3E ..0.O~..G8.C...>
1630 28 CD 51 04 CD 95 0E ED 5B 66 01 D5 3E 2C CD 51 (.Q.....[f..>,.Q
1640 04 CD 95 0E ED 5B 66 01 D5 01 FF 00 7E FE 2C 20 .....[f.....~.,
1650 15 C5 CD 18 17 C1 1D 43 1C 28 AA 7E FE 2C 20 06 .......C.(.~., .
1660 C5 CD 18 17 C1 4B 3E 29 CD 51 04 E3 C5 CD 66 15 .....K>).Q....f.
1670 C1 EB E1 E3 D5 C5 CD 66 15 C1 D1 78 96 D2 D1 16 .......f...x....
1680 ED 44 B9 30 01 4F 23 23 7E 23 66 6F E5 C5 48 06 .D.0.O##~#fo..H.
1690 00 09 C1 EB 79 96 38 38 3C 4F 46 23 23 7E 23 66 ....y.88<OF##~#f
16A0 6F EB 78 B7 28 1D C5 06 00 1A ED B1 79 C1 20 20 o.x.(.......y.
16B0 4F C5 D5 E5 18 06 13 1A BE 20 03 23 10 F8 E1 D1 O......... .#....
16C0 C1 20 09 D1 ED 52 7D CD 32 12 E1 C9 79 B7 20 D6 . ...R}.2...y. .
16D0 D1 AF 18 F3 CD 1E 17 4F ED 78 C3 32 12 CD 0A 17 .......O.x.2....
16E0 ED 79 C9 CD 0A 17 47 C5 1E 00 CD 48 09 28 08 3E .y....G....H.(.>
16F0 2C CD 51 04 CD 1B 17 C1 ED 78 AB A0 28 FA C9 EB ,.Q......x..(...
1700 3E 29 CD 51 04 C1 D1 C5 43 C9 CD 1B 17 D5 3E 2C >).Q....C.....>,
1710 CD 51 04 CD 1B 17 C1 C9 CD 47 09 CD 85 0E CD 27 .Q.......G.....'
1720 0A 7A B7 C2 44 0A CD 48 09 7B C9 CD 9D 15 CA 0E .z..D..H.{......
1730 20 5F 23 23 7E 23 66 6F E5 19 46 72 E3 C5 7E CD  _##~#fo..Fr..~.
1740 DE 23 C1 E1 70 C9 38 09 CD 1B 03 EE 03 4F C3 1E .#..p.8......O..
1750 03 CD 1B 17 FE 04 D2 6D 04 47 CD 1B 03 E6 FC B0 .......m.G......
1760 18 EB CD 9F 18 CD 1B 17 FE 0E DA 44 0A FD 77 01 ...........D..w.
1770 4F D6 0E 30 FC C6 1C ED 44 81 FD 77 02 C9 CD 14 O..0....D..w....
1780 0A DA 6D 04 4F 06 03 CD 0C 03 38 1B 3C 20 F6 10 ..m.O.....8.< ..
1790 F6 CD 0C 03 38 11 3C 28 F8 3D B9 28 05 CD AB 17 ....8.<(.=.(....
17A0 18 E5 0E 07 C3 0F 03 1E 13 18 11 06 03 CD B6 17 ................
17B0 B7 20 F8 10 F8 C9 CD 0C 03 D0 1E 14 C3 7E 04 14 . ...........~..
17C0 C4 B4 05 18 F5 CD 14 0A DA 6D 04 4F CD 47 09 C2 .........m.O.G..
17D0 6D 04 E5 C5 01 FF 08 CD 12 03 10 FB C1 CD 12 03 m...............
17E0 2A 5E 01 EB 2A 5C 01 4E 23 CD 12 03 7C 92 20 02 *^..*\.N#...!. .
17F0 7D 93 20 F3 01 FF 08 CD 12 03 10 FB E1 C9 FE 97 }. .............
1800 16 FF 28 0B 32 66 01 CD B4 05 16 00 21 65 01 CD ..(.2f......!e..
1810 47 09 CD 7E 17 2A 5C 01 06 03 CD 0C 03 38 A0 5F G..~.*\......8._
1820 96 A2 20 18 73 CD 3E 04 7E B7 23 20 EB 10 EB 22 .. .s.>.~.# ..."
1830 5E 01 3A AB 01 B7 CC DB 0C C3 56 05 1E 15 C3 7E ^.:.......V....~
1840 04 C0 E1 01 FF 08 CD 12 03 10 FB 2A 5C 01 7E 23 ...........*\.~#
1850 B6 23 28 31 5E 23 56 23 E5 CD 18 12 CD DD 24 23 .#(1^#V#......$#
1860 CD 95 18 E1 7E FE 09 28 05 0E 20 CD 12 03 CD 37 ....~..(.. ....7
1870 08 E5 21 B1 01 CD 95 18 E1 0E 0D CD 12 03 0E 0A ..!.............
1880 CD 12 03 18 C9 0E 1A CD 12 03 01 FF 08 CD 12 03 ................
1890 10 FB C3 A0 04 7E B7 C8 4F CD 12 03 23 18 F6 EB .....~..O...#...
18A0 21 FC 05 E3 E5 EB FD 21 15 03 FD 22 8E 01 FD 21 !......!..."...!
18B0 95 01 C9 08 EB 2A 54 01 7C A5 3C C2 44 0A 2A 5C .....*T.!.<.D.*\
18C0 01 7E 23 B6 EB CA 0D 0B 11 0A 00 D5 08 DC 85 09 .~#.............
18D0 ED 53 A1 01 D1 FE 2C 20 06 CD 47 09 DC 85 09 ED .S...., ..G.....
18E0 53 9F 01 E5 2A 5C 01 23 23 5E 23 56 E1 FE 2C 20 S...*\.##^#V..,
```

Abb. 4-1

```
18F0 0E CD 47 09 DC 85 09 2A A1 01 ED 52 DA 6D 04 ED ..G....*...R.m..
1900 53 AC 01 B7 C2 6D 04 CD 92 05 D2 E1 0A 60 69 D9 S....m........`i.
1910 11 01 00 21 00 00 D9 11 FF FF CD 95 05 D9 2B EB ...!...........+.
1920 ED 4B 9F 01 78 B1 CA 6D 04 CD C6 23 ED 5B A1 01 .K..x..m...#.[..
1930 19 DA 6B 11 2A 5C 01 23 23 4E 23 46 EB 2A AC 01 ..k.*\.##N#F.*..
1940 ED 42 38 06 28 0E 60 69 18 0D ED 4B 9F 01 2A 9A .B8.(.`i...K..*.
1950 01 09 18 03 2A A1 01 22 9A 01 EB CD 47 09 B7 CA ....*.."....G...
1960 68 1A FE 88 28 14 FE 8C 28 10 FE A2 28 0C FE C9 h...(...(...(...
1970 28 08 FE 9D 28 04 FE 8B 20 E1 CD 47 09 30 DF 22 (...(... ..G.0."
1980 9C 01 2B 11 00 00 0E 00 CD 47 09 30 2B E5 B7 21 ..+......G.0+..!
1990 98 19 ED 52 30 11 21 89 2E CD BD 07 CD 9A 1A E1 ...R0.!.........
19A0 CD 47 09 38 FB 18 B7 62 6B 29 29 19 29 D6 30 5F .G.8...bk)).).0_
19B0 16 00 19 EB E1 0C 18 D0 79 32 9E 01 E5 2A AC 01 ........y2...*..
19C0 EB E5 ED 52 38 2A CD 92 05 60 69 D1 D9 2A A1 01 ...R8*...`i..*..
19D0 ED 5B 9F 01 D9 CD 95 05 38 13 21 75 2E D5 CD BD .[......8.!u....
19E0 07 E1 CD C9 24 CD 9A 1A E1 7E C3 5E 19 D9 E5 D9 ....$....~.^....
19F0 D1 E1 AF 06 98 CD AB 22 CD DD 24 06 00 23 E5 7E ......."..$..#.~
1A00 B7 28 04 04 23 18 F8 3A 9E 01 90 28 3C 38 1C 4F .(..#..:...(<8.O
1A10 06 00 2A 9C 01 54 5D 09 E5 2A 5E 01 ED 52 ED 42 ..*..T]..*^..R.B
1A20 44 4D E1 ED B0 ED 53 5E 01 18 1E ED 44 4F 06 00 DM....S^....DO..
1A30 2A 5E 01 54 5D 09 CD 3E 04 22 5E 01 EB E5 ED 4B *^.T]..>."^....K
1A40 9C 01 ED 42 44 4D E1 ED B8 D1 2A 9C 01 1A B7 28 ...BDM....*....(
1A50 05 77 23 13 18 F7 E5 2A 5C 01 EB CD 5F 05 E1 7E .w#....*\..._..~
1A60 FE 2C CA 7A 19 C3 5E 19 23 B6 23 B6 2B C2 37 19 .,.z..^.#.#.+.7.
1A70 ED 5B AC 01 CD 92 05 60 69 ED 4B 9F 01 ED 5B A1 .[.....`i.K...[.
1A80 01 23 23 73 23 72 23 7E B7 20 FB EB 09 EB 23 B6 .##s#r#~. ....#.
1A90 23 B6 20 EE CD C6 05 C3 A0 04 2A 9A 01 CD C1 24 #. .......*....$
1AA0 C3 EE 0C CD 73 05 D1 C5 CD 92 05 D2 44 0A 54 5D ....s.......D.T]
1AB0 E3 E5 7C 92 20 02 7D 93 D2 44 0A CD DB 0C C1 21 ..|. .}..D.....!
1AC0 56 05 E3 EB 2A 5E 01 1A 02 03 13 7C 92 20 02 7D V...*^.....|. .}
1AD0 93 20 F4 ED 43 5E 01 C9 3E 7F 01 3E BF F5 CD 9A . ..C^..>..>....
1AE0 0E CD 48 09 20 14 CD BF 22 3D 2F 47 F1 4F 2F A0 ..H. ..."=/G.O/.
1AF0 47 3A A3 01 A1 B0 32 A3 01 C9 F1 C3 6D 04 CD 85 G:....2......m...
1B00 09 C0 E1 CD 92 05 D2 44 0A 60 69 23 23 4E 23 46 .......D.`i##N#F
1B10 23 C5 CD 37 08 E1 E5 CD C9 24 CD 64 07 21 B1 01 #..7.....$.d.!..
1B20 E5 1E FF 1C 7E E6 7F 77 23 20 F8 E1 57 06 00 CD ....~..w# ..W...
1B30 C9 07 FE 3A 30 10 FE 30 38 0C D6 30 4F 78 07 07 ...:0..08..0Ox..
1B40 80 07 81 47 18 E9 E5 21 2D 1B E3 05 04 20 01 04 ...G...!-.... ..
1B50 D9 21 71 1B BE 23 4E 23 46 23 C5 D9 C8 D9 C1 34 .!q..#N#F#.....4
1B60 35 20 F1 FE 60 38 04 E6 5F 18 E6 D9 3E 07 C3 72 5 ..`8.._...>..r
1B70 07 20 A4 1B 51 C7 09 4C DC 1B 46 B5 1B 49 16 1C . ..Q..L..F..I..
1B80 44 E6 1B 0D 65 1C 52 FF 1B 45 68 1C 58 11 1C 4B D...e.R..Eh.X..K
1B90 AF 1B 48 0E 1C 7F 59 1C 41 9C 1B 00 C1 D1 CD EE ..H...Y.A.......
1BA0 0C C3 03 1B 7E B7 C8 14 CD 66 07 23 10 F6 C9 E5 ....~....f.#....
1BB0 21 FA 1B E3 37 F5 CD C9 07 4F F1 35 34 C8 F5 DC !...7....O.54...
1BC0 FA 1B 7E CD 66 07 F1 F5 30 05 CD 75 1C 18 02 23 ..~.f...0..u...#
1BD0 14 7E B7 28 05 B9 20 EB 10 E9 F1 C9 CD 2A 08 CD .~.(.. .......*..
1BE0 EE 0C C1 C3 15 1B 7E B7 C8 3E 5C CD 72 07 7E B7 ......~..>\.r.~.
1BF0 28 08 CD 66 07 CD 75 1C 10 F4 3E 5C C3 72 07 7E (..f..u...>\.r.~
1C00 B7 C8 CD C9 07 CD 66 07 77 23 14 10 F2 C9 36 00 .......f.w#....6.
1C10 5A 06 FF CD A4 1B CD C9 07 FE 0D 28 48 FE 18 C8 Z...........(H...
1C20 FE 7F 20 0F 15 14 28 12 2B 15 7E CD 66 07 CD 75 .. ...(.+.~.f..u
1C30 1C 18 E3 F5 7B FE FF 38 08 F1 3E 07 CD 72 07 18 ....{..8..>..r..
1C40 D5 92 1C 14 D5 EB 6F 26 00 19 44 4D 03 CD 1F 04 .......o&..DM....
1C50 D1 F1 CD 66 07 77 23 18 BD 7A B7 C8 15 2B 7E CD ...f.w#..z...+~.
1C60 66 07 10 F5 C9 CD 2A 08 CD EE 0C C1 D1 37 F5 21 f.....*.......7.!
1C70 B1 01 C3 F9 04 E5 1D 7E B7 28 07 23 7E 2B 77 23 ........~.(.#~+w#
1C80 18 F5 E1 C9 C0 EB 21 87 01 ED 5F 77 23 77 23 77 .......!..._w#w#w
1C90 23 77 23 E6 7F 77 23 36 80 EB C9 CD 9F 18 C0 E5 #w#..w#6.........
1CA0 2A 5E 01 CD EE 0C CD 93 09 ED 5B 60 01 7C 92 20 *^.........[`.|.
1CB0 02 7D 93 28 3B 4E 23 7E 23 CB 7F 20 2B CD 72 07 .}.(;N#~#.. +.r.
1CC0 79 E6 7F C4 72 07 3E 24 CB 79 C4 72 07 3E 3D CD y...r.>$.y.r.>=.
```

Abb. 4-1

```
1CD0 72 07 79 17 9F 22 66 01 E5 20 09 CD EC 22 CD DD r.y.."f.. ...".. 
1CE0 24 CD ED 13 CD 34 14 E1 23 23 23 23 23 23 18 B3 $....4..######..
1CF0 E1 C9 CD B4 05 2A 5E 01 2B CD 64 1D 38 27 28 F9 .....*^.+.d.8'(.
1D00 FE 7F 28 F5 23 77 CD 3E 04 7E D6 0D 20 13 77 23 ..(.#w.>.~.. .w#
1D10 77 23 77 23 77 CD 64 1D 38 0B FE 0A 28 DB B7 18 w#w#w.d.8...(...
1D20 DD FE 0D 20 D4 36 00 23 36 1A 2A 5E 01 E5 7E D6 ... .6.#6.*^..~.
1D30 1A 20 09 32 AB 01 CD C2 05 C3 A0 04 7E 23 B7 20 . .2.........~#.
1D40 FB 7E 23 B7 28 FB 2B 22 9A 01 E1 3E FF 32 AB 01 .~#.(.+"...>.2..
1D50 CD 48 09 3C 3D CA A9 04 30 05 F5 AF C3 F1 04 1E .H.<=...0.......
1D60 18 C3 7E 04 CD 0C 03 D8 E6 7F C9 CD B4 05 21 B0 ..~...........!.
1D70 01 06 00 CD 64 1D 38 32 28 F9 FE 7F 28 F5 FE 1A ....d.82(...(...
1D80 28 28 FE 0A 20 04 04 05 28 E9 4F 78 FE FF 79 28 ((.. ...(.Ox..y(
1D90 02 23 04 77 D6 0D 20 DB 77 3E FE 32 AB 01 21 B0 .#.w.. .w>.2..!.
1DA0 01 CD 47 09 3C 3D 28 C6 18 AE CD C2 05 C3 A0 04 ..G.<=(.........
1DB0 32 66 01 CD 47 09 11 FF FF 28 0E 3E 2C CD 51 04 2f..G....(.>,.Q.
1DC0 D2 6D 04 CD 85 09 C2 6D 04 ED 53 9A 01 3E FD 32 .m.....m..S..>.2
1DD0 AB 01 C3 07 18 3C CA 2D 1D 3C 28 92 CD C2 05 AF .....<.-.<(.....
1DE0 32 AB 01 ED 5B 9A 01 7A A3 3C 01 CC 08 C5 C8 C3 2...[..z.<......
1DF0 DA 0A C8 D2 44 0A CD 85 09 ED 53 9A 01 11 0A 00 ....D.....S.....
1E00 FE 2C 20 06 CD 47 09 DC 85 09 ED 53 9C 01 3E AD ., ..G.....S..>.
1E10 CD 51 04 CD 73 05 D1 C5 60 69 D9 11 01 00 21 00 .Q..s...`i....!.
1E20 00 D9 CD 95 05 D2 44 0A E5 D9 EB ED 4B 9C 01 78 ......D.....K..x
1E30 B1 CA 6D 04 CD C6 23 ED 5B 9A 01 19 DA 6B 11 E5 ..m...#.[....k..
1E40 ED 5B 9A 01 CD 92 05 DA 44 0A D1 C5 60 69 7E 23 .[......D...`i~#
1E50 B6 28 0A 23 7E 23 66 6F ED 52 DA 6B 11 C1 E1 D1 .(.#~#fo.R.k....
1E60 E5 ED 52 E3 2B ED 42 EB 38 0A ED 42 DA 44 0A 09 ..R.+.B.8..B.D..
1E70 EB E1 E5 19 E3 E5 50 59 ED 4B 5E 01 09 E5 CD 19 ......PY.K^.....
1E80 04 E1 22 5E 01 60 69 C1 E3 D5 ED B0 E1 ED 4B 9A .."^.`i.......K.
1E90 01 23 23 71 23 70 23 7E B7 20 FB 23 D1 7C 92 20 .##q#p#~. .#.|.
1EA0 02 7D 93 28 0B D5 EB 2A 9C 01 09 44 4D EB 18 E1 .}.(...*...DM...
1EB0 CD DB 0C C3 56 05 CD 6A 10 D5 E5 21 6D 01 CD 19 ....V..j...!m...
1EC0 23 2A 60 01 E3 3A 02 01 F5 3E 2C CD 51 04 CD 6A #*`..:...>,.Q..j
1ED0 10 C1 3A 02 01 A8 1F DA 90 0E E3 EB E5 2A 60 01 ..:..........*`.
1EE0 7C 92 20 02 7D 93 C2 44 0A D1 E1 E3 D5 CD 19 23 |. .}..D.......#
1EF0 E1 11 6D 01 CD 19 23 E1 C9 3E 01 32 4E 01 C3 6A ..m...#..>.2N..j
1F00 10 E5 19 EB 2A 62 01 B7 ED 52 E3 C1 EB 1B 1B 1B ....*b...R......
1F10 1B 28 02 ED B0 ED 53 62 01 32 4E 01 E1 7E FE 2C .(....Sb.2N..~.,
1F20 C0 CD 47 09 18 D3 C8 ED 73 9A 01 E5 11 57 1F D5 ..G.....s....W..
1F30 CD 24 0A 0E 00 D5 CD 48 09 28 10 3E 2C CD 51 04 .$.....H.(.>,.Q.
1F40 0C C5 CD 24 0A C1 EB E3 EB 18 EA D5 EB 2A 9A 01 ...$.........*..
1F50 2B 72 2B 73 2B 2B C9 E1 C9 3E 00 C4 1B 17 C0 FE +r+s++...>......
1F60 0B 28 F6 D2 44 0A 32 AF 01 C9 CD 27 0A 1A C3 32 .(..D.2....'...2
1F70 12 CD 24 0A D5 3E 2C CD 51 04 CD 1B 17 D1 12 C9 ..$..>,.Q.......
1F80 21 08 2A CD 05 23 18 0C CD 05 23 18 04 C1 DD E1 !.*..#....#.....
1F90 D1 CD D2 22 78 B7 C8 3A 6B 01 B7 CA F5 22 90 30 ..."x..:k....".0
1FA0 11 ED 44 D9 DD E5 CD 02 23 D9 DD E3 CD F5 22 D9 ..D.....#.....".
1FB0 DD E1 FE 29 D0 F5 CD 21 23 67 F1 CD 8F 20 B4 21 ...)...!#g... .!
1FC0 66 01 F2 D6 1F CD 5A 20 30 69 23 34 CA 55 20 2E f.....Z 0i#4.U .
1FD0 01 CD AC 20 18 5D AF 90 47 7E 9B 5F 23 7E 9A 57 ... .]..G~._#~.W
1FE0 23 7E DD 9D DD 6F 23 7E DD 9C DD 67 23 7E 99 4F #~...o#~...g#~.O
1FF0 DC 72 20 68 63 AF 47 79 B7 20 27 DD 4C DD 7D DD .r hc.Gy. '.L.}.
2000 67 DD 6A AF 54 65 6F 78 D6 08 FE D0 20 E8 AF 32 g.j.Teox.... ..2
2010 6B 01 C9 05 29 CB 12 08 DD 29 08 30 02 DD 23 08 k...).....).0..#.
2020 CB 11 F2 13 20 78 5C 45 B7 28 08 21 6B 01 86 77 .... x\E.(.!k...w
2030 30 DC C8 78 21 6B 01 B7 FC 45 20 46 23 7E E6 80 0..x!k...E F#~..
2040 A9 4F C3 F5 22 1C C0 14 C0 DD 2C C0 DD 24 C0 0C .O.."......,..$..
2050 C0 0E 80 34 C0 1E 06 C3 7E 04 7E 83 5F 23 7E 8A ...4....~.~._#~.
2060 57 23 7E DD 8D DD 6F 23 7E DD 8C DD 67 23 7E 89 W#~...o#~...g#~.
2070 4F C9 21 6C 01 7E 2F 77 AF 6F 67 90 47 7D ED 52 O.!l.~/w.og.G}.R
2080 EB 6F DD 9D DD 6F 7D DD 9C DD 67 7D 99 4F C9 06 .o...o}...g}.O..
2090 00 D6 08 38 10 43 5A DD 55 08 DD 7C DD 6F 08 DD ...8.CZ.U..|.o..
20A0 61 0E 00 18 EC C6 09 6F AF 2D C8 79 1F 4F DD 7C a......o.-.y.O.|
```

Abb. 4-1

```
20B0 1F DD 67 DD 7D 1F DD 6F CB 1A CB 1B CB 18 18 E8 ..g.}..o........
20C0 00 00 00 00 00 81 06 23 85 AC C3 11 7F 53 CB 9E .......#.....S..
20D0 B7 23 7F CC FE A6 0D 53 7F CB 5C 60 BB 13 80 DD .#.....S..\`....
20E0 E3 4E 38 76 80 5C 29 3B AA 38 82 CD BF 22 B7 EA .N8v.\);.8..."..
20F0 44 0A 21 6B 01 7E 01 35 80 DD 21 F3 04 11 FA 33 D.!k.~.5..!....3
2100 90 F5 70 D5 DD E5 C5 CD 94 1F C1 DD E1 D1 04 CD ..p.............
2110 C3 21 21 C0 20 CD 88 1F 21 C6 20 CD DB 2A 01 80 .!!. ...!. ..*..
2120 80 DD 21 00 00 11 00 00 CD 94 1F F1 CD A1 24 01 ..!...........$.
2130 31 80 DD 21 17 72 11 D2 F7 18 04 C1 DD E1 D1 CD 1..!.r..........
2140 BF 22 C8 2E 00 CD 63 22 79 D5 D9 4F D1 DD E5 E1 ."....c"y..O....
2150 D9 01 00 00 50 58 DD 21 00 00 21 F3 1F E5 21 68 ....PX.!..!...!h
2160 21 E5 E5 E5 E5 21 66 01 7E 23 B7 20 0E 43 5A DD !....!f.~#. .CZ.
2170 55 08 DD 7C DD 6F 08 DD 61 4F C9 E5 EB 1E 08 1F U..|.o..aO......
2180 57 79 30 12 E5 D9 E3 19 E3 EB DD E5 E3 ED 5A E3 Wy0...........Z.
2190 DD E1 EB 89 D9 E1 1F 4F DD 7C 1F DD 67 DD 7D 1F .......O.|..g.}.
21A0 DD 6F CB 1C CB 1D CB 18 1D 7A 20 D3 EB E1 C9 CD .o.......z .....
21B0 DA 22 01 20 34 DD 21 00 00 11 00 00 CD F5 22 C1 .". ..!........".
21C0 DD E1 D1 CD BF 22 CA 70 04 2E FF CD 63 22 FD E5 ....."p....c"..
21D0 34 34 2B E5 D9 E1 4E 2B 56 2B 5E 2B 7E 2B 6E 67 44+...N+V+^+~+ng
21E0 EB D9 41 EB DD E5 FD E1 AF 4F 57 5F DD 21 00 00 ..A......OW_.!..
21F0 32 AE 01 E5 FD E5 C5 E5 78 D9 E3 B7 ED 52 E3 EB 2.......x....R..
2200 FD E5 E3 ED 52 E3 FD E1 EB 99 D9 E1 47 3A AE 01 ....R.......G:..
2210 DE 00 3F 30 09 32 AE 01 F1 F1 F1 37 18 04 C1 FD ..?0.2.....7....
2220 E1 E1 0C 0D 1F FA 5E 22 17 CB 13 CB 12 08 DD 29 ......^".......)
2230 08 30 02 DD 23 08 CB 11 29 08 FD 29 08 30 02 FD .0..#...)..).0..
2240 23 08 CB 10 3A AE 01 17 32 AE 01 79 B2 B3 DD B4 #...:...2..y....
2250 DD B5 20 9F E5 21 6B 01 35 E1 20 97 18 2D FD E1 .. ..!k.5. ..-..
2260 C3 34 20 78 B7 28 20 7D 21 6B 01 AE 80 47 1F A8 .4 x.( }!k...G..
2270 78 F2 86 22 C6 80 77 CA AD 21 CD 21 23 77 2B C9 x.."..w..!.!#w+.
2280 CD BF 22 2F B7 21 B7 E1 F2 0E 20 C3 55 20 CD 02 .."/.!.... .U ..
2290 23 78 B7 C8 C6 02 38 F3 47 CD 94 1F 21 6B 01 34 #x....8.G...!k.4
22A0 C0 18 E8 CD BF 22 06 88 11 00 00 21 6B 01 4F D5 ....."......!k.O.
22B0 DD E1 11 00 00 70 06 00 23 36 80 17 C3 F0 1F 3A .....p..#6.....:
22C0 6B 01 B7 C8 3A 6A 01 FE 2F 17 9F C0 3C C9 CD BF k...:j../...<...
22D0 22 F0 21 6A 01 7E EE 80 77 C9 EB 2A 66 01 E3 E5 ".!j.~..w..*f...
22E0 2A 68 01 E3 E5 2A 6A 01 E3 E5 EB C9 11 66 01 01 *h...*j......f..
22F0 06 00 ED B0 C9 ED 53 66 01 DD 22 68 01 ED 43 6A ......Sf.."h..Cj
2300 01 C9 21 66 01 5E 23 56 23 4E DD 69 23 4E DD 61 ..!f.^#V#N.i#N.a
2310 23 4E 23 46 23 C9 11 66 01 01 06 00 EB ED B0 EB #N#F#..f........
2320 C9 21 6A 01 7E 07 37 1F 77 3F 1F 23 23 77 79 07 .!j.~.7.w?.##wy.
2330 37 1F 4F 1F AE C9 78 B7 28 85 CD BF 22 79 28 88 7.O...x.(..."y(.
2340 21 6A 01 AE 79 FA C8 22 CD 50 23 1F A9 C3 C8 22 !j..y..".P#...."
2350 23 78 BE C0 2B 79 BE C0 2B DD 7C BE C0 2B DD 7D #x..+y..+.|..+.}
2360 BE C0 2B 7A BE C0 2B 7B 96 C0 E1 C9 5E 23 56 23 ..+z..+{....^#V#
2370 4E 23 46 23 C9 47 4F 57 5F DD 67 DD 6F B7 C8 E5 N#F#.GOW_.g.o...
2380 CD 02 23 CD 21 23 AE 67 F2 9B 23 1B 7A A3 3C 20 ..#.!#.g..#.z.<
2390 0A DD 2B DD 7C DD A5 3C 20 01 0D 3E A8 90 CD 8F ..+.|..< ..>....
23A0 20 7C 17 DC 45 20 06 00 DC 72 20 E1 C9 21 6B 01  |..E ...r ..!k.
23B0 7E FE A8 3A 66 01 D0 7E CD 75 23 36 A8 7B F5 79 ~..:f..~.u#6.{.y
23C0 17 CD F0 1F F1 C9 21 00 00 78 B1 C8 3E 11 3D C8 ......!..x..>.=.
23D0 29 38 08 EB 29 EB 30 F6 09 30 F3 C3 6B 11 FE 26 )8..).0..0..k..&
23E0 CA 71 24 FE 2D F5 28 05 FE 2B 28 01 2B CD 0E 20 .q$.-.(..+(.+..
23F0 47 57 5F 2F 4F CD 47 09 38 56 FE 2E 28 26 FE 45 GW_/O.G.8V..(&.E
2400 20 25 CD 47 09 15 FE A6 28 0E FE 2D 28 0A 14 FE  %.G....(..-(...
2410 2B 28 05 FE A5 28 01 2B CD 47 09 38 49 14 20 07 +(...(.+.G.8I. .
2420 AF 93 5F 0C 0C 28 CE E5 7B 90 F2 39 24 CD DA 22 .._..(..{..9$.."
2430 ED 44 21 C0 20 CD EC 22 37 F5 CD AE 24 20 FB F1 .D!. .."7...$ ..
2440 30 07 C1 DD E1 D1 CD C3 21 D1 F1 CC D2 22 EB C9 0.......!....."..
2450 D5 57 78 89 47 C5 E5 D5 CD 8E 22 F1 D6 30 CD A1 .Wx.G....."..0..
2460 24 E1 C1 D1 18 8F 7B 87 87 83 87 86 D6 30 5F 18 $.....{.......0_.
2470 A7 11 00 00 CD 47 09 28 22 38 0C D6 41 38 1C FE .....G.("8..A8..
2480 06 30 18 C6 0A 18 02 D6 30 08 7A FE 10 D2 55 20 .0......0.z...U
```

Abb. 4-1

```
2490 08 EB 29 29 29 29 B5 6F EB 18 D9 E5 CD 18 12 E1 ..)))).o........
24A0 C9 CD DA 22 CD A6 22 C1 DD E1 D1 C3 94 1F C8 F5 ..."..".........
24B0 CD 8E 22 F1 3D C9 D5 E5 F5 CD AF 21 F1 E1 D1 3C ..".=......!...<
24C0 C9 E5 21 93 2F CD BD 07 E1 CD 17 12 21 AF 01 7E ..!./.......!..~
24D0 36 00 F5 CD DD 24 F1 32 AF 01 C3 30 14 AF 32 AA 6....$.2...0..2.
24E0 01 21 6E 01 36 20 E6 08 28 02 36 2B CD BF 22 F2 .!n.6 ..(.6+..".
24F0 FA 24 36 2D E5 CD D2 22 E1 B4 23 36 30 3A AA 01 .$6-..."..#60:..
2500 57 17 DA B2 25 CA AA 25 E5 CD 1F 27 21 AF 01 34 W...%..%...'!..4
2510 35 28 51 57 C6 0B FA 55 25 BE 28 02 30 37 47 7E 5(QW...U%.(.07G~
2520 90 3C 4F 04 7A 16 0B E1 23 CD 83 26 E5 AF 01 00 .<O.z...#..&....
2530 00 ED B1 2B 01 3C 25 C5 AF F5 18 47 E1 7E FE 2D ...+.<%....G.~.-
2540 C8 FE 20 C8 FE 30 28 0A FE 25 20 05 23 7E FE 2D .. ..0(..% .#~.-
2550 C8 2B 36 20 C9 4E 0D 28 01 0C 06 02 E1 23 7A 16 .+6 .N.(.....#z.
2560 00 C3 D7 26 01 00 03 C6 0C FA 74 25 FE 0D 30 04 ...&......t%..0.
2570 3C 47 3E 02 D6 02 E1 F5 CD 5A 27 36 30 20 01 23 <G>......Z'60 .#
2580 CD 6D 27 2B 7E FE 30 28 FA FE 2E 28 01 23 F1 28 .m'+~.0(...(.#.(
2590 1A 36 45 23 36 2B F2 9D 25 36 2D ED 44 06 2F 04 .6E#6+..%6-.D./.
25A0 D6 0A 30 FB C6 3A 23 70 23 77 23 36 00 EB 21 6E ..0..:#p#w#6..!n
25B0 01 C9 23 C5 E5 7A 1F DA CE 26 01 0E B6 DD 21 C9 ..#..z...&....!.
25C0 1B 11 04 BF CD 36 23 FA D3 25 E1 C1 CD DD 24 2B .....6#..%....$+
25D0 36 25 C9 16 0B CD BF 22 C4 1F 27 E1 C1 FA 83 26 6%.....".'....&
25E0 C5 5F 78 92 93 F4 F9 27 CD 01 28 CD 6D 27 B3 C4 ._x....'..(.m'..
25F0 1B 28 83 C4 5A 27 D1 7B B7 20 01 2B 3D F4 F9 27 .(..Z'.{. .+=..'
2600 E5 21 6E 01 46 0E 20 3A AA 01 5F E6 20 28 07 78 .!n.F. :.._. (.x
2610 B9 0E 2A 20 01 41 71 CD 47 09 28 10 FE 45 28 0C ..* .Aq.G.(..E(.
2620 FE 30 28 F2 FE 2C 28 EE FE 2E 20 03 2B 36 30 CB .0(..,(... .+60.
2630 63 28 03 2B 36 24 CB 53 20 02 2B 70 E1 28 02 70 c(.+6$.S .+p.(.p
2640 23 36 00 21 6D 01 23 3A A8 01 95 92 C8 7E FE 20 #6.!m.#:.....~.
2650 28 F4 FE 2A 28 F0 2B E5 F5 CD 47 09 FE 2D 28 F8 (..*(.+...G..-(.
2660 FE 2B 28 F4 FE 24 28 F0 FE 30 20 10 23 CD 47 09 .+(..$(..0 .#.G.
2670 30 0A 2B 01 2B 77 F1 28 FB C1 18 CB F1 28 FD E1 0.+.+w.(.....(..
2680 36 25 C9 5F 79 B7 28 01 3D 83 FA 8E 26 AF C5 F5 6%._y.(.=...&...
2690 FC B6 24 20 FB C1 7B 90 C1 5F 82 78 FA AA 26 92 ..$ ..{.._.x..&.
26A0 93 F4 F9 27 C5 CD 01 28 18 11 CD F9 27 79 CD 5D ...'...(....'y.]
26B0 27 4F AF 92 93 CD F9 27 C5 47 4F CD 6D 27 C1 B1 'O.....'.GO.m'..
26C0 20 03 2A A8 01 83 3D F4 F9 27 50 C3 00 26 CD BF  .*...=..'P..&..
26D0 22 37 C4 1F 27 E1 C1 F5 79 B7 F5 28 01 3D 80 4F "7..'...y..(.=.O
26E0 7A E6 04 FE 01 9F 57 81 4F D6 0B F5 C5 FC B6 24 z.....W.O......$
26F0 FA ED 26 C1 F1 C5 F5 FA FB 26 AF ED 44 80 3C 82 ..&......&..D.<.
2700 47 0E 00 CD 6D 27 F1 F4 15 28 C1 F1 20 01 2B F1 G...m'...(.. .+.
2710 38 04 C6 0B 90 92 C5 CD 91 25 EB D1 C3 00 26 D5 8........%....&.
2720 AF F5 CD 48 27 01 15 A2 DD 21 F8 02 11 FD FF CD ...H'....!......
2730 36 23 F2 45 27 F1 CD AF 24 F5 C3 25 27 F1 CD B6 6#.E'...$..%'...
2740 24 F5 CD 48 27 F1 D1 C9 01 3A A5 DD 21 B7 43 11 $..H'....:..!.C.
2750 FC 3F CD 36 23 E1 F2 3D 27 E9 05 20 08 36 2E 22 .?.6#..='.. .6."
2760 A8 01 23 48 C9 0D C0 36 2C 23 0E 03 C9 D5 C5 E5 ..#H...6,#......
2770 CD 80 1F 3C CD 75 23 CD F5 22 E1 C1 11 C2 27 3E ...<.u#.."....'>
2780 0B CD 5A 27 C5 F5 E5 D5 CD 02 23 E1 06 2F 04 7B ..Z'......#../.{
2790 96 5F 23 7A 9E 57 23 DD 7D 9E DD 6F 23 DD 7C 9E ._#z.W#.}..o#.|.
27A0 DD 67 23 79 9E 4F 2B 2B 2B 2B 30 E2 CD 5A 20 23 .g#y.O++++0..Z #
27B0 CD F5 22 EB E1 70 23 F1 C1 3D 20 C5 CD 5A 27 77 .."..p#..= ..Z'w
27C0 D1 C9 00 E4 0B 54 02 00 CA 9A 3B 00 00 E1 F5 05 .....T....;.....
27D0 00 80 96 98 00 00 40 42 0F 00 00 A0 86 01 00 00 ......@B........
27E0 10 27 00 00 00 E8 03 00 00 00 64 00 00 00 00 0A .'........d.....
27F0 00 00 00 00 01 00 00 00 00 B7 C8 3D 36 30 23 18 ...........=60#.
2800 F8 7B 82 3C 47 3C D6 03 30 FC C6 05 4F 3A AA 01 .{.<G<..0...O:..
2810 E6 40 C0 4F C9 20 04 C8 CD 5A 27 36 30 23 3D 18 .@.O. ...Z'60#=.
2820 F6 CD 47 09 30 1E CD 85 09 E5 CD 92 05 D2 E1 0A ..G.0...........
2830 60 69 23 23 23 CD 47 09 D6 9C C2 44 0A 47 CD F0 `i###.G....D.G..
2840 13 E1 18 03 CD 95 0E CD 48 09 37 28 0C FE 2C 28 ........H.7(..,(
2850 05 FE 3B C2 6D 04 CD 47 09 EB 2A 66 01 01 D1 EB ..;.m..G..*f....
2860 E5 F5 D5 46 B0 CA 44 0A 23 23 4E 23 66 69 C3 77 ...F..D.##N#fi.w
```

Abb. 4-1

```
2870 28 CD F9 29 CD 72 07 AF 5F 57 CD F9 29 57 7E 23 (..).r.._W..)W~#
2880 FE 23 CA BE 28 FE 27 CA 81 29 05 CA 6D 29 FE 2B .#..(.'..)..m).+
2890 3E 08 28 E6 2B 7E 23 FE 2E CA D6 28 BE 20 D2 FE >.(.+~#....(. ..
28A0 24 28 14 FE 2A 20 CA 78 FE 02 23 38 03 7E FE 24 $(..* .x..#8.~.$
28B0 3E 20 20 07 05 1C FE AF C6 10 23 1C 82 57 1C 0E >  .......#..W..
28C0 00 05 28 46 7E 23 FE 2E 28 17 FE 23 28 F0 FE 2C ..(F~#..(..#(..,
28D0 20 19 CB F2 18 E8 7E FE 23 3E 2E C2 71 28 0E 01  .....~.#>..q(..
28E0 23 0C 05 28 25 7E 23 FE 23 28 F6 D5 54 5D FE 5E #..(%~#.#(..T].^
28F0 20 16 BE 20 13 23 BE 20 0F 23 BE 20 0B 23 78 D6  .. .#. .#. .#x.
2900 04 38 05 D1 47 14 23 CA EB D1 2B 1C CB 5A 20 13 .8..G.#...+..Z .
2910 1D 78 B7 28 0E 7E D6 2D 28 06 FE FE 20 05 CB DA .x.(.~.-(... ...
2920 CB D2 05 E1 F1 28 51 C5 D5 CD 9A 0E D1 C1 C5 E5 .....(Q.........
2930 43 78 81 FE 19 D2 44 0A 7A F6 80 CD DE 24 CD 31 Cx....D.z....$.1
2940 14 E1 CD 48 09 37 28 0C FE 3B 28 05 FE 2C C2 6D ...H.7(..;(..,.m
2950 04 CD 47 09 C1 EB E1 E5 F5 D5 7E 90 23 23 4E 23 ..G.......~.##N#
2960 66 69 16 00 5F 19 78 B7 C2 77 28 18 06 CD F9 29 fi.._.x..w(....)
2970 CD 72 07 E1 F1 C2 5E 28 DC EE 0C E3 CD 66 15 E1 .r....^(.....f..
2980 C9 0E 01 1E 4C 05 28 1C 7E 23 FE 45 28 0C FE 52 ....L.(.~#.E(..R
2990 28 08 FE 4C 28 04 FE 43 20 0A 5F 0C 05 28 05 7E (..L(..C ._..(.~
29A0 23 BB 28 F7 CD F9 29 E1 F1 28 CD C5 D5 CD 95 0E #.(...)..(......
29B0 D1 C1 C5 E5 2A 66 01 41 0E 00 78 FE 45 28 2B D5 ....*f.A..(.E(+.
29C0 C5 CD CD 15 C1 D1 78 96 47 7B FE 4C 28 0E FE 52 ......x.G{.L(..R
29D0 28 15 78 CB 38 90 08 CD F1 29 08 47 C5 CD 34 14 (.x.8....).G..4.
29E0 C1 CD F1 29 C3 41 29 AF 18 EC 78 96 30 ED AF 18 ...).A)...x.0...
29F0 EA 04 05 C8 CD 64 07 18 F9 F5 7A B7 3E 2B C4 72 .....d....z.>+.r
2A00 07 F1 C9 21 D2 22 E3 E9 00 00 00 00 00 80 CD DA ...!."..........
2A10 22 21 08 2A CD EC 22 C1 DD E1 D1 CD BF 22 28 44 "!.*.."......"(D
2A20 78 B7 CA 0F 20 D5 DD E5 C5 79 F6 7F CD 02 23 F2 x... ....y....#.
2A30 44 2A D5 DD E5 C5 CD AD 23 C1 DD E1 D1 F5 CD 36 D*......#......6
2A40 23 E1 7C 1F E1 22 6A 01 E1 22 68 01 E1 22 66 01 #.|.."j.."h.."f.
2A50 DC 03 2A CC D2 22 D5 DD E5 C5 CD EB 20 C1 DD E1 ..*.."...... ...
2A60 D1 CD 3F 21 01 38 81 DD 21 3B AA 11 5C 29 CD 3F ..?!.8..!;..\).?
2A70 21 3A 6B 01 FE 88 D2 80 22 CD DA 22 CD AD 23 C1 !:k....."..".#.
2A80 DD E1 D1 F5 CD 91 1F 21 9E 2A CD EA 2A 21 6B 01 .......!.*..*!k.
2A90 F1 B7 FA 97 2A 86 01 86 3F 77 D0 C3 80 22 0A CC ....*...?w...".. 
2AA0 D5 45 56 15 6A CF 37 A0 92 27 6D F5 95 EE 93 00 .EV.j.7..'m.....
2AB0 71 D0 FC A7 78 21 74 B1 21 82 C4 2E 77 82 58 58 q...x!t.!...w.XX
2AC0 95 1D 7A 6D CB 46 58 63 7C E9 FB EF FD 75 7E D2 ..zm.FXc|....u~.
2AD0 F7 17 72 31 80 00 00 00 00 00 81 CD DA 22 11 3B ..r1.........".;
2AE0 21 D5 E5 CD 02 23 CD 3F 21 E1 CD DA 22 7E 23 CD !....#.?!..."~#.
2AF0 EC 22 FE F1 C1 DD E1 D1 3D C8 D5 DD E5 C5 F5 E5 .".....=........
2B00 CD 3F 21 E1 CD 05 23 E5 CD 94 1F E1 18 E5 CD BF .?!...#.........
2B10 22 FA 35 2B 21 87 01 CD EC 22 C8 01 35 98 DD 21 ".5+!...."..5..!
2B20 7A 44 11 00 00 CD 3F 21 01 28 68 DD 21 46 B1 11 zD....?!.(h.!F..
2B30 00 00 CD 94 1F CD 02 23 7B 59 4F 36 80 2B 46 36 .......#{YO6.+F6
2B40 80 CD F3 1F 21 87 01 C3 16 23 21 9C 2B CD 83 1F ....!....#!.+...
2B50 CD DA 22 01 49 83 DD 21 DA 0F 11 21 A2 CD F5 22 ..".I..!...!..."
2B60 C1 DD E1 D1 CD C3 21 CD DA 22 CD AD 23 C1 DD E1 ......!.."..#...
2B70 D1 CD 91 1F 21 A2 2B CD 88 1F CD BF 22 37 F2 88 ....!.+....."7..
2B80 2B CD 80 1F CD BF 22 B7 F5 F4 D2 22 21 A2 2B CD +....."...."!.+.
2B90 83 1F F1 D4 D2 22 21 A8 2B C3 DB 2A 21 A2 DA 0F ....."!.+..*!...
2BA0 49 81 00 00 00 00 00 7F 07 90 BA 34 76 6A 82 E4 I..........4vj..
2BB0 E9 E7 4B F1 84 B1 4F 7F 3B 28 86 31 B6 64 69 99 ..K...O.;(.1.di.
2BC0 87 E4 36 E3 35 23 87 24 31 E7 5D A5 86 21 A2 DA ..6.5#.$1.]..!..
2BD0 0F 49 83 CD DA 22 CD 50 2B C1 DD E1 D1 CD DB 22 .I...".P+......"
2BE0 EB CD F5 22 CD 4A 2B C3 BF 21 CD BF 22 FC 03 2A ...".J+..!.."..*
2BF0 FC D2 22 3A 6B 01 FE 81 38 10 01 00 81 DD 21 00 ..":k...8.....!.
2C00 00 51 59 CD C3 21 21 88 1F E5 21 14 2C CD DB 2A .QY..!!...!.,..*
2C10 21 9C 2B C9 0D 14 07 BA FE 62 75 51 16 CE D8 D6 !.+......buQ....
2C20 78 4C BD 7D D1 3E 7A 01 CB 23 C4 D7 7B DC 3A 0A xL.}.>z..#..{.:.
2C30 17 34 7C 36 C1 A3 81 F7 7C EB 16 61 AE 19 7D 5D .4|6....|..a..}]
2C40 78 8F 60 B9 7D A2 44 12 72 63 7D 16 62 FB 47 92 x.`.}.D.rc}.b.G.
```

Abb. 4-1

```
2C50 7E C0 F0 BF CC 4C 7E 7E 8E AA AA AA 7F F6 FF FF ~....L~~........
2C60 FF 7F 80 45 4E C4 46 4F D2 4E 45 58 D4 44 41 54 ...EN.FO.NEX.DAT
2C70 C1 49 4E 50 55 D4 44 49 CD 52 45 41 C4 4C 45 D4 .INPU.DI.REA.LE.
2C80 47 4F 20 54 CF 46 4E 45 4E C4 49 C6 52 45 53 54 GO T.FNEN.I.REST
2C90 4F 52 C5 47 4F 20 53 55 C2 52 45 54 55 52 CE 52 OR.GO SU.RETUR.R
2CA0 45 CD 53 54 4F D0 4F 55 D4 4F CE 4E 55 4C CC 57 E.STO.OU.O.NUL.W
2CB0 41 49 D4 44 45 C6 50 4F 4B C5 50 52 49 4E D4 BF AI.DE.POK.PRIN..
2CC0 4C 49 53 54 45 CE 43 4C 45 41 D2 46 4E 52 45 54 LISTE.CLEA.FNRET
2CD0 55 52 CE 53 41 56 C5 A1 55 53 49 4E C7 54 41 42 UR.SAV..USIN.TAB
2CE0 A8 54 CF 46 CE 53 50 43 A8 54 48 45 CE 4E 4F D4 .T.F.SPC.THE.NO.
2CF0 53 54 45 D0 AB AD AA AF DE 41 4E C4 4F D2 BE BD STE......AN.O...
2D00 BC 53 47 CE 49 4E D4 41 42 D3 55 53 D2 46 52 C5 .SG.IN.AB.US.FR.
2D10 49 4E D0 50 4F D3 53 51 D2 52 4E C4 4C 4F C7 45 IN.PO.SQ.RN.LO.E
2D20 58 D0 43 4F D3 53 49 CE 54 41 CE 41 54 CE 50 45 X.CO.SI.TA.AT.PE
2D30 45 CB 4C 45 CE 53 54 52 A4 56 41 CC 41 53 C3 43 E.LE.STR.VA.AS.C
2D40 48 52 A4 4C 45 46 54 A4 52 49 47 48 54 A4 4D 49 HR.LEFT.RIGHT.MI
2D50 44 A4 4C 50 4F D3 49 4E 53 54 D2 45 4C 53 C5 4C D.LPO.INST.ELS.L
2D60 50 52 49 4E D4 54 52 41 43 C5 4C 54 52 41 43 C5 PRIN.TRAC.LTRAC.
2D70 52 41 4E 44 4F 4D 49 5A C5 53 57 49 54 43 C8 4C RANDOMIZ.SWITC.L
2D80 57 49 44 54 C8 4C 4E 55 4C CC 57 49 44 54 C8 4C WIDT.LNUL.WIDT.L
2D90 56 41 D2 4C 4C 56 41 D2 53 50 45 41 CB A7 50 52 VA.LLVA.SPEA..PR
2DA0 45 43 49 53 49 4F CE 43 41 4C CC 4B 49 4C CC 45 ECISIO.CAL.KIL.E
2DB0 58 43 48 41 4E 47 C5 4C 49 4E C5 4C 4F 41 44 47 XCHANG.LIN.LOADG
2DC0 CF 52 55 CE 4C 4F 41 C4 4E 45 D7 41 55 54 CF 43 .RU.LOA.NE.AUT.C
2DD0 4F 50 D9 41 4C 4F 41 44 C3 41 4D 45 52 47 45 C3 OP.ALOAD.AMERGE.
2DE0 41 4C 4F 41 C4 41 4D 45 52 47 C5 41 53 41 56 C5 ALOA.AMERG.ASAV.
2DF0 4C 49 53 D4 4C 4C 49 53 D4 52 45 4E 55 4D 42 45 LIS.LLIS.RENUMBE
2E00 D2 44 45 4C 45 54 C5 45 44 49 D4 43 4F 4E D4 00 .DELET.EDI.CON..
2E10 7C 53 4E 45 58 54 20 57 2F 4F 20 46 4F D2 53 59 !SNEXT W/O FO.SY
2E20 4E 54 41 58 20 45 52 52 4F D2 52 45 54 55 52 4E NTAX ERRO.RETURN
2E30 20 57 2F 4F 20 47 4F 53 55 C2 4F 55 54 20 4F 46  W/O GOSU.OUT OF
2E40 20 44 41 54 C1 49 4C 4C 45 47 41 4C 20 46 55 4E  DAT.ILLEGAL FUN
2E50 43 54 49 4F CE 41 52 49 54 48 4D 45 54 49 43 20 CTIO.ARITHMETIC 
2E60 4F 56 45 52 46 4C 4F D7 4F 55 54 20 4F 46 20 4D OVERFLO.OUT OF M
2E70 45 4D 4F 52 D9 55 4E 44 45 46 49 4E 45 44 20 53 EMOR.UNDEFINED S
2E80 54 41 54 45 4D 45 4E 54 A0 53 55 42 53 43 52 49 TATEMENT.SUBSCRI
2E90 50 54 20 4F 55 54 20 4F 46 20 52 41 4E 47 C5 52 PT OUT OF RANG.R
2EA0 45 2D 44 49 4D 45 4E 53 49 4F 4E 45 44 20 41 52 E-DIMENSIONED AR
2EB0 52 41 D9 43 41 4E 27 54 20 2F B0 49 4C 4C 45 47 RA.CAN'T /.ILLEG
2EC0 41 4C 20 44 49 52 45 43 D4 54 59 50 45 20 4D 49 AL DIREC.TYPE MI
2ED0 53 2D 4D 41 54 43 C8 4E 4F 20 53 54 52 49 4E 47 S-MATC.NO STRING
2EE0 20 53 50 41 43 C5 53 54 52 49 4E 47 20 54 4F 4F  SPAC.STRING TOO
2EF0 20 4C 4F 4E C7 54 4F 4F 20 43 4F 4D 50 4C 45 D8  LON.TOO COMPLE.
2F00 43 41 4E 27 54 20 43 4F 4E 54 49 4E 55 C5 55 4E CAN'T CONTINU.UN
2F10 44 45 46 49 4E 45 44 20 55 53 45 52 20 43 41 4C DEFINED USER CAL
2F20 CC 46 49 4C 45 20 4E 4F 54 20 46 4F 55 4E C4 49 .FILE NOT FOUN.I
2F30 4C 4C 45 47 41 4C 20 45 4F C6 46 49 4C 45 53 20 LLEGAL EO.FILES 
2F40 44 49 46 46 45 52 45 4E D4 52 45 43 4F 56 45 52 DIFFEREN.RECOVER
2F50 45 C4 46 4E 52 45 54 55 52 4E 20 57 2F 4F 20 46 E.FNRETURN W/O F
2F60 55 4E 43 54 49 4F 4E 20 43 41 4C CC 4D 49 53 53 UNCTION CAL.MISS
2F70 49 4E 47 20 53 54 41 54 45 4D 45 4E 54 20 4E 55 ING STATEMENT NU
2F80 4D 42 45 D2 2A 49 4E 56 41 4C 49 44 20 49 4E 50 MBE.*INVALID INP
2F90 55 54 8A 20 40 20 4C 49 4E 45 A0 0A 52 45 41 44 UT. @ LINE..READ
2FA0 59 3A 8A 2A 45 58 54 52 41 20 4C 4F 53 54 8A 0A Y:.*EXTRA LOST..
2FB0 2A 42 52 45 41 CB 00 CD EE 0C 2A 10 2E CD C9 24 *BREA.....*....$
2FC0 CD EE 0C 21 B5 30 CD BD 07 AF 21 00 01 3F 77 2C ...!.0....!..?w,
2FD0 20 FC 24 38 F8 2B F9 22 5A 01 21 FF FF 22 54 01  .$8.+."Z.!.."T.
2FE0 3E 2C 32 B0 01 3E C3 32 00 00 32 8D 01 32 A4 01 >,2..>.2..2..2..
2FF0 21 C9 2F 22 A5 01 FD 21 90 01 21 0F 03 22 8E 01 !./"...!..!.."..
3000 3E 48 32 91 01 32 96 01 3E 38 32 92 01 32 97 01 >H2..2..>82..2..
3010 21 03 03 22 01 00 21 08 01 22 06 01 21 00 80 22 !.."..!.."..!.."
3020 87 01 22 89 01 22 8B 01 21 EF 30 CD BD 07 CD E2 .."..".."!.0.....
```

Abb. 4-1

```
3030 06 CD 47 09 FE 41 28 8B 2F FE AC CA B7 2F 3C 20 ..G..A(./..../<
3040 07 CD 21 03 60 6F 18 14 21 B1 01 CD 48 09 CD DE ..!.`o..!...H...
3050 23 7E B7 C2 6D 04 CD 27 0A EB 2B 2B 22 04 01 22 #~..m..'..++".."
3060 48 01 E5 11 00 03 B7 ED 52 E1 DA 4D 04 11 9C FF H.......R..M....
3070 19 11 00 03 7C 92 20 02 7D 93 11 B7 2F 30 03 11 ....|. .}../0..
3080 00 03 7C 92 20 02 7D 93 38 E0 F9 22 5A 01 EB 22 ..|. .}.8.."Z.."
3090 5C 01 CD 3E 04 B7 EB ED 52 01 F0 FF 09 CD EE 0C \..>....R.......
30A0 CD C9 24 21 FE 30 CD BD 07 21 BD 07 22 A5 01 CD ..$!.0...!.."...
30B0 B4 05 C3 A0 04 54 2E 44 2E 4C 2E 20 5A 2D 38 30 .....T.D.L. Z-80
30C0 20 42 41 53 49 43 20 62 79 20 4E 65 69 6C 20 43  BASIC by Neil C
30D0 6F 6C 76 69 6E 20 26 20 52 6F 67 65 72 20 41 6D olvin & Roger Am
30E0 69 64 6F 6E 0A 4D 61 79 20 20 31 39 37 37 8A 0A idon.May  1977..
30F0 48 69 67 68 65 73 74 20 4D 65 6D 6F 72 F9 20 42 Highest Memor. B
3100 79 74 65 73 20 46 72 65 65 0A 0A 57 65 6C 63 6F ytes Free..Welco
3110 6D 65 20 74 6F 20 42 41 53 49 43 2C 20 56 65 72 me to BASIC, Ver
3120 2E 20 32 2E 31 0A 3C 54 44 4C 20 5A 2D 38 30 20 . 2.1.<TDL Z-80
3130 48 69 67 68 20 50 72 65 63 69 73 69 6F 6E 20 45 High Precision E
3140 78 74 65 6E 64 65 64 20 56 65 72 73 69 6F 6E 3E xtended Version>
3150 8A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
31A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
31B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
31C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
31D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
31E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
31F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3200 04 .
```

Abb. 4-1

4.1 Anpassung des Basic-Interpreters

```
300  C3  xx  xx  BASIC:   JMP  INIT
303  C3  xx  xx  REST:    JMP  RECOVER
306  C3  xx  xx  USR:     JMP  ERROR     ;vorbelegt mit Fehler
309  C3  03  F0  CI:      JMP  CIN       ;Konsolen-Eingabe
30C  C3  06  F0  RI:      JMP  RIV       ;Leser-Eingabe
30F  C3  09  F0  CO:      JMP  CON       ;Konsolen-Ausgabe
312  C3  0C  F0  PO:      JMP  WRTV      ;Stanzer-Ausgabe
315  C3  0F  F0  LO:      JMP  LISTX     ;List-Ausgabe
318  C3  12  F0  CSTS:    JMP  CSTSX     ;Konsolen-Status

31B  C3  15  F0  IOCHK:   JMP  IOCHX     ;I/O-Konfig.-Test

31E  C3  18  F0  IOSET:   JMP  IOSTX     ;I/O-Modifikation
321  C3  1B  F0  MEMSIZ:  JMP  MEMCK     ;Speicherende-Test
324  C3  1E  F0  TRAP:    JMP  TRAPX     ;Breakpoint
```

Das 12-K-Basic benötigt einen RAM-Speicher von 100H bis 2FFH. Der Speicher darunter bleibt unbelegt. Das Basic ist von Adresse 300H ab in ein EPROM schreibbar.

CIN	Holt ein Zeichen von der Konsole in den Akku.
RIV	Holt ein Zeichen vom Leser in den Akku.
CON	Im C-Register steht ein Zeichen für die Ausgabe auf Konsole.
WRTV	Ausgabe eines Zeichens über Register C an den Stanzer (PUNCH).
LISTX	Ausgabe eines Zeichens über C an die LIST-Device, z.B. zur Ausgabe von Basic-Programmen.
CSTSX	Konsolstatus. Übergibt in A den Wert FFH, wenn ein Zeichen da ist, ohne dieses einzulesen, sonst 0.
IOCHX	Im A-Register wird die aktuelle Konfiguration übergeben. Kann auch mit XRA A RET kurzgeschlossen werden.
IOSTX	Das neue I/O-Byte wird im C-Register übergeben. Kann ebenfalls kurzgeschlossen werden.
MEMCK	Im Register B steht die höherwertige Adresse in A die niederwertige Adresse des höchsten für das Basic-Programm zur Verfügung stehenden Speicherplatzes. Nach Starten wird die Speichergröße gefragt; wird mit CR geantwortet, so wird der hier angegebene Wert verwendet.
TRAPX	Starten des eigenen Monitorprogramms.

4.2 12-KByte-Basic-Befehlsbeschreibung

1. Allgemeine Dienstprogramme

AUTO
Mit dem Befehl AUTO wird die automatische Erzeugung von Zeilennummern veranlaßt. Dabei kann noch eine Startzeile und die Schrittweite angegeben werden, z.B. AUTO 100 oder AUTO 100,3.

CLEAR
Alle Variablen werden gelöscht. Wird zusätzlich eine Zahl angegeben, so wird der Stringplatz auf diesen Wert gesetzt. CLEAR 200 löscht alle Variablen und setzt den verfügbaren Platz für Strings auf 200.

CONTINUE
Wurde der Programmlauf mit CTRL-C gestoppt, so kann mit diesem Befehl die Ausführung fortgesetzt werden, wenn keine Veränderungen am Programm vorgenommen wurden.

DELETE
Ein Bereich von Zeilen kann gelöscht werden. DELETE 100–135 löscht alle Zeilen von 100 bis einschließlich 135.

KILL
Damit kann nicht verwendeter Speicherplatz von Matrizen zurückgewonnen werden. KILL A,B entfernt den Speicherplatz, den die Matrizen A und B verbraucht haben.

LOAD
Lädt ein Programm vom Leser. Es wird dabei zuvor NEW durchgeführt. LOAD P lädt ein Programm, das mit SAVE P abgespeichert wurde. Dabei ist nur ein Buchstabe als Name zulässig. LOAD? P führt ein Prüflesen durch. Wird die Datei geladen,

so wird das ASCII-Zeichen „Bell" ausgegeben.

LOADGO
Wie LOAD, aber mit Starten des Programms. LOADGO P,100 lädt das Programm P und startet dieses auf 100.

NEW
Löscht den gesamten Arbeitsspeicher, d.h. das Basic-Programm und alle Variablen.

PRECISION
Voreingestellt sind 11 Digits Rechengenauigkeit. PRECISION 4 veranlaßt z.B., daß Zahlen vor der Ausgabe auf 4 Stellen gerundet werden. Intern wird aber weiterhin mit 11 Digits gerechnet.

RENUMBER
Neunumerierung. Es werden alle Referenzen, Sprünge usw. automatisch umgestellt. RENUMBER numeriert von 10 beginnend in 10er Schritten alle Zeilen. RENUMBER 110 beginnt bei 110. RENUMBER 120,3 numeriert beginnend mit 120 in 3er-Schritten. RENUMBER 500,5,300 beginnt erst bei der Zeile 300 mit einer Neunumerierung, die dann ab 500 in 5er Schritten läuft. Damit können Lücken eingebaut werden.

RUN
Löscht alle Variablen und startet die Programmausführung. Eine angegebene Zeilennummer bewirkt den Start von da ab.

SAVE
Abspeichern eines Basic-Programms über den Stanzer (Punch). SAVE P speichert ein vorhandenes Basic-Programm unter dem Namen P. Nur ein Buchstabe ist als Name zulässig.

2. Der EDIT-Befehl

EDIT
Mit Edit kann eine Programmzeile verbessert werden. EDIT 10 korrigiert Zeile 10, falls vorhanden.

Dazu gibt es weitere Befehle, die aus Buchstaben und Zahlen bestehen, die nicht über die Konsole ausgegeben werden. Zahlen reichen von 1 bis 255 und werden hier mit einem kleinen n gekennzeichnet:

A	Lädt den EDIT-Buffer vom Programmspeicher neu.
nD	n Zeichen werden gelöscht (delete).
E	Beende EDIT und ersetze die Zeile.
nFx	Finde das n-te Zeichen x im Buffer und halte den Pointer direkt davor an.
H	Lösche alles rechts des Pointers und gehe in den INSERT-Modus.
I	Füge alle folgenden Zeichen ein, bis CR oder ESC eingegeben wird.
nKx	Lösche Zeichen vom Pointer bis zum n-ten Zeichen x, lösche dieses aber nicht.
L	Ausgabe der Zeile (list).
Q	Verlasse EDIT ohne Ersetzung (quit).
nR	Ersetze die folgenden n Zeichen durch n vorhandene (replace).
X	Pointer ans Zeilenende und INSERT.
Space	Pointer nach rechts.
Rubout	Pointer nach links.
cr (Return)	Ende des EDIT mit Ersetzung.
Escape	Ende des INSERT Modes.

3. Befehle für die Konsole

LIST
Ausgabe eines Programms. LIST 10–100 gibt alle Zeilen von 10 bis 100 aus, LIST 20– gibt von 20 an alle Zeilen aus.

LVAR
Ausgabe aller aktuellen Variablen und deren Belegung.

NULL
Für langsame Konsolen: NULL 3,255 sorgt dafür, daß nach jedem CR LF drei ASCII-Zeichen FFH (Rubout) gesendet werden.

POS
Die aktuelle Position der Konsoldruckstelle wird übergeben, z.B. A=POS (B), B ist ein „dummy"-Wert.

PRINT
Ausgabebefehl. Beispiel:
PRINT 123,A, „TEST";B,CS. Bei Angabe eines Kommas wird alle 14 Spalten positioniert, bei Strichpunkten werden zwei Leerzeichen gelassen. Steht am Schluß ein Komma oder Strichpunkt, so wird kein CR/LF ausgegeben.

PRINT USING
Für PRINT USING gibt es zwei Formen.
PRINT USING zeile; ausgabe liste
PRINT USING string; ausgabe liste
Im zweiten Beispiel wird die Formatierung durch die Stringvariable bestimmt. Bei Angabe einer Zeile muß in dieser mit einem „!" beginnend das Format abgelegt sein.

#	Numerisches Feld.
.	Dezimalpunkt-Position.
+	Kann am Ende oder am Anfang einer numerischen Spezifikation stehen.
–	Entsprechend dem Pluszeichen; nur werden positive Zahlen mit einem führenden Leerzeichen ausgegeben.
**	Leere Positionen werden mit * gefüllt.
$$	$ wird unmittelbar vor die erste Ziffer gestellt.
**$	Kombination beider Zeichen.
,	Ein Komma links des Dezimalpunktes zeigt an, daß Kommas alle drei Stellen eingefügt werden.
↑↑↑↑	Vier dieser Zeichen bestimmen, daß die Zahl in Exponentialschreibweise ausgegeben wird.

Stringfelder werden mit ' eingeleitet. Sie können durch ein oder mehrere der folgenden Zeichen gefolgt werden:

L	Linksangleich
R	Rechtsangleich
C	Zentrierung
E	Linksangleich mit Erweiterung, falls der String zu lang ist.

SPC
Damit kann eine Zahl von Leerzeichen ausgegeben werden.

PRINT A;SPC(5);B es werden 5 zusätzliche Leerzeichen zwischen A und B gefügt.

SWITCH
Damit kann die Konsolzuweisung geändert werden. Dazu erhält SWITCH ein Argument zwischen 0 und 3.

0 = TTY
1 = CRT
2 = BATCH USE
3 = USER DEFINED

(Siehe auch Zapple-Monitor, [5]).

TAB
Damit kann direkt eine bestimmte Druckposition erreicht werden.

PRINT A;TAB(30);B B wird beginnend bei der Position 30 ausgegeben.

TRACE
TRACE 1 schaltet den Trace-Modus ein, TRACE 0 schaltet ihn aus. Anstelle der Zahl kann auch ein arithmetischer Ausdruck stehen. Ist dieser ungleich 0, so wird eingeschaltet. Es werden dann alle ausgeführten Zeilennummern in „<>“ ausgegeben.

WIDTH
Damit kann die Ausgabelänge angegeben werden, bei der automatisch ein CRLF eingefügt wird. WIDTH 80 stellt die Länge auf 80 Zeichen pro Zeile ein; Minimum ist 15, Maximum 255.

4. Befehle für den Zeilendrucker (LIST-Device)

Die meisten der vorhergehenden Befehle können auch auf einen Drucker geschaltet werden, wenn folgende Befehle verwendet werden:

LLIST
LLVAR
LNULL
LPRINT
LPRINT USING
LTRACE
LWIDTH
LPOS
SPC
TAB

5. Befehle und Funktionen, die Daten transportieren

LET
Zuweisung eines Wertes an eine (stets links von „=“ stehende) Variable.

Beispiel: 10 LET A=20
LET kann aber auch weggelassen werden und dient nur der Übersicht.

Beispiele:
200 DIM A(10),B(40)
210 DIM C(50,10)
220 DIM D(J)
230 DIM A$(221)

DATA
Konstantenablage, die durch READ geholt werden kann, z.B.
10 DATA 5,8,7,9,1.4

READ
Holen der Konstanten, die in DATA (s.o.) angegeben sind.
20 READ A
Beim ersten Mal erhält A hier den Wert 5, dann 8 usw. bis 1.4; danach führt ein erneuter Aufruf zum Fehler.

RESTORE
Damit kann der Lesezeiger von READ auf den Anfang gesetzt werden. Bei diesem Basic-Interpreter kann durch Angabe einer Zeilennummer zusätzlich der Zeiger auf eine bestimmte Zeile gesetzt werden, z.B.
200 RESTORE 30

LINE INPUT
Damit kann eine ganze Zeile in eine Stringvariable eingelesen werden.
Allgemeine Form: LINE INPUT „prompt string“; Eingabe Liste „prompt string“ ist optional.

INPUT
Eingabe von Daten über die Konsole.
Beispiel:
10 INPUT „GEBEN SIE A,B,C ein“;A,B,C
20 INPUT B$

INP
Lesen eines Z80-Ports.
A=INP(0)
A erhält den Wert des Kanals 0.

OUT
OUT 1,7 gibt den Wert 7 an den Kanal 1.

DIM
Reservieren von Speicherplatz für Matrizen. Matrizen können 1 bis 255 Dimensionen haben.

WAIT
Automatische Warteschleife für Ports.
WAIT A,B,C
Der Wert am Port A wird mit C exklusiv-ODER verknüpft sowie mit B UND-verknüpft. Erst wenn das Ergebnis nicht mehr Null ist, wird das Basic-Programm fortgesetzt.

PEEK
Direktspeicherzugriff: B=PEEK(A) holt den dezimalen Inhalt der Speicherzelle mit der dezimalen Adresse A in B.

POKE
Mit POKE A,B wird B in die Zeile mit der Adresse A geschrieben.

COPY
Damit können Teile des Basic-Programms verschoben oder dupliziert werden.
COPY neue Zeile, increment=Zeilenbereich
Die im Zeilenbereich angegebenen Zeilen werden auf den neuen Bereich kopiert und dabei umnumeriert.

EXCHANGE
Schneller Austausch von Variablen-Werten.
EXCHANGE A$,B$
EXCHANGE C,D(I,J)
Im Falle von Strings werden intern nur Pointer vertauscht.

6. Steuerung des Programmflusses

GOTO
Sprunganweisung (Argument = Zeilennummer).

GOSUB
Unterprogrammaufruf (Argument = Zeilennummer).

RETURN
Unterprogrammrücksprung.

ON x GOTO
ON x GOSUB
Es wird auf die x-te Zeilennummer gesprungen.
10 ON A GOTO 100,125,145
Ist A=1, so wird auf 100 gesprungen. Falls A gleich 0 ist oder größer als die Anzahl der angegebenen Zeilennummern, so wird die nächste Zeile ausgeführt.

CALL
Maschinenunterprogrammaufruf.
CALL adresse, argument 1,...,argument n.
Ruft ein Programm beginnend bei „adresse" auf. Jedes der Argumente wird in eine 16-Bit-Zahl gewandelt und nach folgendem Schema übergeben:

```
SP->   argument n
       ...
       argument 1
HL->   Return-Adresse
BC->   Zahl der Argumente auf dem
       Stack
```

FOR, TO, STEP, NEXT
Schleifenanweisung, die beliebig geschachtelt werden kann:

```
10 FOR A=B TO C STEP D
20 ...
30 NEXT A
```

IF, THEN, ELSE
Bedingte Anweisung, wobei hier in Ergänzung zum Standard-Basic auch ELSE verwendet werden kann.

Beispiel:
10 IF B=4 THEN 50 ELSE 30

Ebenfalls möglich
20 IF Z$=„JA" GOTO 50
30 IF G=5 THEN G=4 ELSE G=7
Als Vergleiche sind zugelassen:

=	Gleich
<>	Ungleich
<	Kleiner
>	Größer
<=	Kleiner Gleich
>=	Größer Gleich

Logische Operatoren:
NOT Negation
AND Und-Verknüpfung
OR Oder-Verknüpfung

Beispiel:
20 IF (A=0) OR NOT (B=4) THEN C=5

7. Trigonometrische Funktionen

ATN
Ermitteln des Arcustangens: A=ATN(.45)
Das Ergebnis wird im Bogenmaß angegeben.

COS
Cosinus berechnen; die Winkelangabe erfolgt im Bogenmaß:
B=COS(3.1415)

SIN
Sinus-Ermittlung: C=SIN(3.1415/2)

TAN
Berechnet den Tangens eines Winkels:
A=TAN(.254)

8. Diverse Funktionen

ABS
Absolutbetrag: ABS(−4.5) ergibt 4.5.

DEF, FN
Damit kann der Benutzer eigene Funktionen definieren. Eine Funktion muß dabei mit FN beginnen, gefolgt von einem Variablennamen, z.B. FNA, FNB6. Der Funktionsname wird von einem in Klammern stehenden Parameter gefolgt.

Beispiel: 200 DEF FNQ(X)=X*B+3
B ist dabei eine Variable, die im Basic-Programm global verwendet wird und X ist lokal auf die Definition beschränkt und stellt nur den Parameter dar.

Beispiel:
```
10 DEF FNA(X)=X*X
. . .
100 A=FNA(3)
110 PRINT A
```
Es wird der Wert 9 ausgegeben.
In diesem Basic wurde die Möglichkeit der DEF-Anweisung beträchtlich erweitert. Es ist möglich, sie über mehrere Zeilen erstrecken zu lassen; außerdem können rekursive Funktionen definiert werden.

Das allgemeine Format lautet:
```
DEF FN name (parameter ,. . .,parameter)
. . .
Funktionskörper
. . .

FNEND funktionswert
```
Es wird im Gegensatz zur Standarddefinition hier einfach das Gleichheitszeichen weggelassen. Die Definition der Funktion wird durch FNEND abgeschlossen, wobei dort der Funktionswert angegeben wird. Die Funktion kann auch vorzeitig abgebrochen werden, indem mit „FNRETURN funktionswert" eine Rückkehr veranlaßt wird. Beispiele:

```
100 DEF FNFAC(I)
200 IF I=0 THEN FNRETURN 1
300 FNEND FNFAC(I−1)*I
400 PRINT FNFAC(6)
```

```
100 DEF FNREP$(I$,I) 'Aufbau eines
    wiederholenden Strings
200 J$=" "
300 IF I<=0 THEN FNRETURN J$
400 FOR J=1 TO I
500 J$=J$+1
600 NEXT
700 FNEND J$
800 PRINT FNREP$("TEST",5)
```

EXP
Exponentialfunktion.
EXP(1) ergibt 2.7182 . . . .

FRE
Damit läßt sich der Restspeicher für Variable und Programm, falls als Dummyparameter eine Variable angegeben wird und der Restspeicher für Strings ermitteln, falls eine Stringvariable als Parameter verwendet wird.
FRE(X) gibt den Variablen und Programmrestspeicher aus.
FRE(X$) den Restspeicher für Strings.

INT
Ermitteln des ganzzahligen Anteils von Zahlen.
20 C=INT(A)
Falls A den Wert 4.56 besitzt, erhält C den Wert 4.

LOG
Ermitteln des natürlichen Logarithmus (d.h. zur Basis e=2,7. . .) einer Zahl.
LOG(EXP(1)) ergibt deshalb den Wert 1.

SGN
Ergibt +1, falls das Argument größer als 0 ist, Null, falls es gleich 0 ist, und −1 , falls das Argument kleiner als 0 ist.
SGN(56.5) ergibt +1.

SQR
Berechnen der Quadratwurzel. Das Argument darf nicht kleiner als 0 sein.
SQR(2) ergibt 1.4142 . . . .

RND
Erzeugen einer Pseudozufallszahl zwischen 0 und 1. Dazu benötigt RND einen Dummy-Parameter. Ist der Wert kleiner 0, wird die RND-Sequenz initialisiert. Ist das Argument 0, so wird der vorhergehende Wert übermittelt. Ein Argument größer als 0 liefert den nächsten RND-Wert in der Sequenz.
10 R=RND(I) 'in R steht eine Zahl zwischen 0 und 1.

RANDOMIZE
Dies ist keine Funktion, vielmehr kann damit ein zufälliger Startpunkt einer Pseudo-Zufallszahlenfolge eingestellt werden.

9. Stringverarbeitung

ASC
Der Dezimalwert eines ersten Zeichens in einem String wird übergeben.
10 A=ASC(A$)
mit A$="A" erhält A den Wert 65.

CHR$
Hier wird gerade umgekehrt das Zeichen übergeben, daß durch den dezimalen Wert des Argument dargestellt wird. Die Codierung erfolgt hierbei in ASCII.
10 A$=CHR$(66)
A$ beinhaltet das Zeichen B.

LEFT$
Erhält zwei Parameter. Der erste Parameter ist ein String. Der zweite Parameter gibt die Anzahl der Zeichen an, die vom linken Rand des Strings an gezählt übergeben werden sollen.

30 B$=LEFT$("STRING",2)
B$ erhält hier die Zeichenfolge "ST".

LEN
Damit kann die Länge eines Strings ermittelt werden:

40 X=LEN(S$)
In X steht die Anzahl der Bytes, die S$ enthält.

MID$
MID$ benötigt drei Parameter. Der erste gibt den String an, der zweite bestimmt die Startposition, der dritte die Anzahl der Zeichen, die von da an verwendet werden sollen.
50 A$=MID$(B$,5,6)
A$ erhält 6 Zeichen, beginnend beim 5ten Zeichen des Strings B$.
MID$ darf auch auf der linken Seite verwendet werden.

RIGHT$
Funktioniert wie LEFT$, nur werden die Zeichen vom rechten Rand hier verwendet.

STR$
Es wird der String übergeben, dessen numerischer Wert in Klammern steht.
10 C$=STR$(7.8)
C$ erhält den String "7.8".

VAL
Gegenteil von STR$: Es wird der numerische Wert eines Strings übergeben.
20 A=VAL("3.4")
A erhält den Wert 3.4. In Basic-Programmen können auch sedezimale Konstanten (Hex-Zahlen) verwendet werden. Sie werden angegeben, indem das Zeichen & vorangestellt wird:

B=&400 'B erhält den Wert 1024.

INSTR
Dient zum Suchen von Strings. Dazu wird als erster Parameter der String angegeben, in dem der zweite Parameter, der Suchstring, gefunden werden soll. Zusätzlich können dann noch eine Startposition und eine Länge angegeben werden.

Beispiele:.

INSTR("123456789","456")	ergibt 4
INSTR("123456789","654")	ergibt 0
INSTR("1234512345","34")	ergibt 3
INSTR("1234512345","34",6)	ergibt 8
INSTR("1234512345","34",6,2)	ergibt 0

10. Sonstige Befehle

END
Beendet die Programmausführung und kann irgendwo innerhalb eines Basic-Programms stehen.

REM
Kennzeichnet, daß die Zeile eine Kommentarzeile ist. Für REM kann auch das Zeichen „ ' " verwendet werden.
10 REM Kommentarzeile
20 A=B 'Kommentar

STOP
Wie END, nur daß BREAK @ LINE . . . ausgegeben wird.

USR
USR erlaubt ein Maschinenunterprogramm aufzurufen. Es wird das Programm über die Sprungadresse bei „USR:" aufgerufen. Um den Parameter zu erhalten, muß das Unterprogramm auf Adresse 300H + 27H gerufen werden. Die Adresse des Parameters steht dann im Registerpaar DE. Um die Information zurück zu übertragen, wird das Unterprogramm auf Adresse 300H + 2AH aufgerufen. Das niederwertige Byte des Ergebnisses wird dazu im Register B übergeben und das höherwertige im Register A. Zur Rückkehr ins Basic-System wird ein RET-Befehl ausgeführt.
10 A=USR(B)
20 PRINT A

11. Befehle für ASCII-Ein/Ausgabe von Programmen

ASAVE
Bei Ausführung des Befehls wird das im Speicher vorhandene Basic-Programm in ASCII-lesbarer Form über den Stanzkanal (TTY) ausgegeben.

ALOAD, ALOADC, AMERGE, AMERGEC
Zum Laden von Programmen, die in ASCII geschrieben sind, also nicht in maschineninterner Form. Jede Zeile muß mit einer Zeilennummer beginnen und mit einem CR beendet sein. Das Einlesen wird entweder durch ein CTRL-Z im Eingabetext oder durch ein EOF des LESE-Unterprogramms beendet. ALOAD löscht ein zuvor vorhandenes Programm, MERGE mischt die ankommenden Zeilen mit dem vorhandenen Programm. Der Unterschied zwischen „A . . ." und „A . . . C" Befehlen liegt in der Art der LESER Behandlung. „A . . .C" setzen einen steuerbaren LESER voraus, und nach der Eingabe einer jeden Zeile wird diese ins Internformat übersetzt. „A . . ."Befehle lesen bis zum Ende und konvertieren erst, nachdem das gesamte Programm eingelesen wurde. Dadurch wird für Programme kurzfristig mehr Speicher benötigt.

12. Steuerzeichen

COMMA	,	Zur nächsten TAB-Position; oder Trennzeichen.
SEMI-COLON	;	Nicht fortschreiten.
COLON	:	Für mehrere Anweisungen in einer Zeile.
RUBOUT	7FH	Löschen des eingegebenen Zeichens in „\\".

CTRL-S	Anhalten der Ausgabe von Daten.
CTRL-Q	Fortfahren bei der Ausgabe.
CTRL-C	Beenden der Ausführung des Basic-Programms.
CTRL-U	Löschen der gerade eingegebenen Zeile.
CTRL-X	Zum Monitorprogramm zurück.
CTRL-O	Unterdrücken der Konsolausgabe.
CTRL-R	Ausgabe der aktuellen Zeile ohne Löschzeichen.
CTRL-T	Während des Programmlaufs kann dadurch die gerade abgearbeitete Zeilennummer ausgegeben werden.

13. Operatoren

In der Reihenfolge ihrer Bearbeitung (hierarchisch geordnet):

( )	Klammer
↑	Exponentialoperator
–	Negation
*	Multiplikation
/	Division
+	Addition
–	Subtraktion
=	Gleichheit
<>	Ungleich
<	Kleiner als
>	Größer als
<=	Kleiner/Gleich
>=	Größer/Gleich
NOT	Logisches NICHT
AND	UND
OR	ODER

4.3 Programmbeispiel ELIZA

ELIZA ist ein Programm, das künstliche Intelligenz simulieren soll. Dieses Programm wurde ursprünglich von Weizenbaum in LISP geschrieben. Es macht aus dem Computer einen Psychologen, der einem Patienten Fragen stellt. *Abb. 4.3-1* zeigt das Listing des Programms. Daß auch schon durch ein so einfaches Programm interessante Gespräche entstehen können, zeigt das Beispiel eines Dialogs in *Abb. 4.3-2*. Das Programm wurde in [8, 9] ausführlich beschrieben.

```
1 REM             E L I Z A
2 REM Modifizierte Version aus Creative Computing
3 REM JUL/AUG 1977
4 REM
5 REM
6 REM BETRIEBSMITTELSTECKBRIEF
7 REM
8 REM COMPUTERTYP:                    Z80 S100 SYSTEM
9 REM PERIFERIE:                      DATENSICHTGERAET
10 REM SPEICHERBEDARF:                12K FUER BASIC U. 10K FUER ELIZA
11 REM BESONDERE BASIC BEFEHLE:
12 REM                                LINE INPUT,RESTORE n,INSTR
13 REM                                MID$,LEFT$,RIGHT$,LEN
14 REM GRENZEN DER ANWENDUNG:         -
15 REM BEMERKUNGEN:                   12K ZAPPLE BASIC VON TDL
16 REM
17 REM
90 CLEAR 500'STRING SPACE FESTLEGEN
100 DIM S(38),R(38),N(38)
110 N1=38:N2=12:N3=115
120 RESTORE 2530
130 FOR X=1 TO N1
140 READ S(X),L: R(X)=S(X): N(X)=S(X)+L-1
150 NEXT X
160 PRINT "HI, I AM ELIZA TELL ME YOUR PROBLEM"
170 LINE INPUT I$ 'EINGABE EINER ZEILE VOM BENUTZER
180 I$=" "+I$+" "
190 REM BESEITIGEN VON APOSTROPHEN
200 FOR L=1 TO LEN(I$)
210 IF MID$(I$,L,1)="'" THEN I$=LEFT$(I$,L-1)+RIGHT$(I$,LEN(I$)-L):GOTO
                                                                     210
220 NEXT L
230 IF I$=P$ THEN PRINT "PLEASE DONT REPEAT YOURSELF":GOTO 170
270 REM SUCHEN DES ERKENNUNGSWORTES
280 RESTORE
290 S=0
300 FOR K=1 TO N1
305 IF S>0 THEN 340
310 READ K$
320 A=INSTR(I$,K$)
330 IF A<>0 THEN S=K:T=A:F$=K$
340 NEXT K
365 IF S>0 THEN K=S:L=T:GOTO 400
370 K=N1:GOTO 570
390 REM KONJUGATIONS PHASE
400 RESTORE 1200
410 C$=" "+RIGHT$(I$,LEN(I$)-LEN(F$)-L+1)
420 FOR X=1 TO N2/2
```

Abb. 4.3-1 Listing von ELIZA

```
430 READ S$,R$
440 FOR L=1 TO LEN(C$)
450 IF L+LEN(S$)>LEN(C$) THEN 510
480 IF MID$(C$,L,LEN(S$))<>S$ THEN 510
490 C$=LEFT$(C$,L-1)+R$+RIGHT$(C$,LEN(C$)-L-LEN(S$)+1)
495 L=L+LEN(R$)
500 GOTO 540
510 IF L+LEN(R$)>LEN(C$) THEN 540
520 IF MID$(C$,L,LEN(R$))<>R$ THEN 540
530 C$=LEFT$(C$,L-1)+S$+RIGHT$(C$,LEN(C$)-L-LEN(R$)+1)
540 NEXT L
550 NEXT X
555 IF MID$(C$,2,1)=" "THEN C$=RIGHT$(C$,LEN(C$)-1)'EIN SPACE IN ANTWORT
560 REM ANTWORTSATZ FINDEN
570 RESTORE 1300
580 FOR X=1 TO R(K):READ F$:NEXT X' POSITIONIEREN
590 R(K)=R(K)+1:IF R(K)>N(K) THEN R(K)=S(K)
600 IF INSTR(F$,"*")=0 THEN PRINT F$:P$=I$:GOTO 170
610 A=INSTR(F$,"*")
620 PRINT LEFT$(F$,A-1);C$;RIGHT$(F$,LEN(F$)-A)
630 GOTO 170
1000 REM KEYWORDS
1050 DATA"CAN YOU","CAN I","YOU ARE","YOURE","I DONT","I FEEL"
1060 DATA "WHY DONT YOU","WHY CANT I","ARE YOU","I CANT","I AM","IM "
1070 DATA"YOU ","I WANT","WHAT","HOW","WHO","WHERE","WHEN","WHY"
1080 DATA"NAME","CAUSE","SORRY","DREAM","HELLO","HI ","MAYBE"
1090 DATA" NO","YOUR","ALWAYS","THINK","ALIKE","YES","FRIEND"
1100 DATA"COMPUTER","ELIZA","MONEY","NOKEYFOUND"
1200 REM KONJUGATIONEN
1230 DATA" ARE "," AM ","WERE ","WAS "," YOU "," I ","YOUR ","MY "
1235 DATA" IVE "," YOUVE "," IM "," YOURE "
1300 REM ANTWORTEN
1330 DATA"DON'T YOU BELIEVE THAT I CAN*"
1340 DATA"PERHAPS YOU WOULD LIKE TO BE ABLE TO*"
1350 DATA"YOU WANT ME TO BE ABLE TO*"
1360 DATA"PERHAPS YOU DON'T WANT TO*"
1365 DATA"DO YOU WANT TO BE ABLE TO*"
1370 DATA"WHAT MAKES YOU THINK I AM*"
1380 DATA "DOES IT PLEASE YOU TO BELIEVE I AM*"
1390 DATA"PERHAPS YOU WOULD LIKE TO BE*"
1400 DATA"DO YOU SOMETIMES WISH YOU WERE*"
1410 DATA"DON'T YOU REALLY*"
1420 DATA"WHY DON'T YOU*"
1430 DATA"DO YOU WISH TO BE ABLE TO*"
1440 DATA"DOES THAT TROUBLE YOU?"
1450 DATA"TELL ME MORE ABOUT SUCH FEELINGS."
1460 DATA"DO YOU OFTEN FEEL*"
1470 DATA"DO YOU ENJOY FEELING*"
1480 DATA"DO YOU REALLY BELIEVE I DON'T*"
1490 DATA"PERHAPS IN GOOD TIME I WILL*"
1500 DATA"DO YOU WANT ME TO*"
1510 DATA"DO YOU THINK YOU SHOULD BE ABLE TO*"
1520 DATA"WHY CAN'T YOU*"
1530 DATA"WHY ARE YOU INTERESTED IN WHETHER OR NOT I AM*"
1540 DATA"WOULD YOU PREFER IF I WHERE NOT*"
1550 DATA"PERHAPS IN YOUR FANTASIES I AM*"
1560 DATA"HOW DO YOU KNOW YOU CAN'T*"
1570 DATA"HAVE YOU TRIED?"
1580 DATA"PERHAPS YOU CAN NOW*"
1590 DATA"DID YOU COME TO ME BECAUSE YOU ARE*"
1600 DATA"HOW LONG HAVE YOU BEEN*"
1610 DATA"DO YOU BELIEVE IT IS NORMAL TO BE*"
```

Abb. 4.3-1

```
1620 DATA"DO YOU ENJOY BEING*"
1630 DATA"WE WERE DISCUSSING YOU-- NOT ME."
1640 DATA"OH, I*"
1650 DATA"YOU'RE NOT REALLY TALKING ABOUT ME, ARE YOU?"
1660 DATA"WHAT WOULD IT MEAN TO YOU IF YOU GOT*"
1670 DATA"WHY DO YOU WANT*"
1680 DATA"SUPPOSE YOU SOON GOT*"
1690 DATA"WHAT IF YOU NEVER GOT*"
1700 DATA"I SOMETIMES ALSO WANT*"
1710 DATA"WHY DO YOU ASK?"
1720 DATA"DOES THAT QUESTION INTEREST YOU?"
1730 DATA"WHAT ANSWER WOULD PLEASE YOU THE MOST?"
1740 DATA"WHAT DO YOU THINK?"
1750 DATA"ARE SUCH QUESTIONS IN YOUR MIND OFTEN?"
1760 DATA"WHAT IS THAT YOY REALLY WANT TO KNOW?"
1770 DATA"HAVE YOU ASKED ANYONE ELSE?"
1780 DATA"HAVE YOU ASKED SUCH QUESTIONS BEFORE?"
1790 DATA"WHAT ELSE COMES TO MIND WHEN YOU ASK THAT?"
1800 DATA"NAMES DON'T INTEREST ME."
1810 DATA"I DON'T CARE ABOUT NAMES-- PLEASE GO ON."
1820 DATA"IS THAT THE REAL REASON?"
1830 DATA"DON'T ANY OTHER REASONS COME TO MIND?"
1840 DATA"DOES THAT REASON EXPLAIN ANYTHING ELSE?"
1850 DATA"WHAT OTHER REASONS MIGHT THERE BE?"
1860 DATA"PLEASE DON'T APOLOGIZE!"
1870 DATA"APOLOGIES ARE NOT NECESSARY."
1880 DATA"WHAT FEELINGS DO YOU HAVE WHEN YOU APOLOGIZE?"
1890 DATA"DON'T BE SO DEFENSIVE!"
1900 DATA"WHAT DOES THAT DREAM SUGGEST TO YOU?"
1910 DATA"DO YOU DREAM OFTEN?"
1920 DATA"WHAT PERSONS APPEAR IN YOUR DREAMS?"
1930 DATA"ARE YOU DISTURBED BY YOUR DREAMS?"
1940 DATA"HOW DO YOU DO ... PLEASE STATE YOUR PROBLEM."
1950 DATA"YOU DON'T SEEM QUITE CERTAIN."
1960 DATA"WHY THE UNCERTAIN TONE?"
1970 DATA"CAN'T YOU BE MORE POSITIVE?"
1980 DATA"YOU AREN'T SURE?"
1990 DATA"DON'T YOU KNOW?"
2000 DATA"WHY NO*"
2010 DATA"DONT SAY NO ITS ALWAYS SO NEGATIVE"
2020 DATA"WHY NOT?"
2030 DATA"ARE YOU SURE?"
2040 DATA"WHY NO?"
2050 DATA"WHY ARE YOU CONCERNED ABOUT MY*"
2060 DATA"WHAT ABOUT YOUR OWN*"
2070 DATA"CAN'T YOU THINK OF A SPECIFIC EXAMPLE?"
2080 DATA"WHEN?"
2090 DATA"WHAT ARE YOU THINKING OF?"
2100 DATA"REALLY, ALWAYS?"
2110 DATA"DO YOU REALLY THINK SO?"
2120 DATA"BUT YOU ARE NOT SURE YOU*"
2130 DATA"DO YOU DOUBT YOU*"
2140 DATA"IN WHAT WAY?"
2150 DATA"WHAT RESEMBLANCE DO YOU SEE?"
2160 DATA"WHAT DOES THE SIMILARITY SUGGEST TO YOU?"
2170 DATA"WHAT OTHER CONNECTIONS DO YOU SEE?"
2180 DATA"COULD THERE REALLY BE SOME CONNECTIONS?"
2190 DATA"HOW?"
2200 DATA"YOU SEEM QUITE POSITIVE."
2210 DATA"ARE YOU SURE?"
2220 DATA"I SEE."
2230 DATA"I UNDERSTAND."
```

Abb. 4.3-1

```
2240 DATA "WHY DO YOU BRING UP THE TOPIC OF FRIENDS?"
2250 DATA"DO YOUR FRIENDS WORRY YOU?"
2260 DATA"DO YOUR FRIENDS PICK ON YOU?"
2270 DATA"ARE YOU SURE YOU HAVE ANY FRIENDS?"
2280 DATA"DO YOU IMPOSE ON YOUR FRIENDS?"
2290 DATA"PERHAPS YOUR LOVE FOR FRIENDS WORRIES YOU."
2300 DATA"DO COMPUTERS WORRY YOU?"
2310 DATA"ARE YOU TALKING ABOUT ME IN PARTICULAR?"
2320 DATA"ARE YOU FRIGHTENED BY MACHINES?"
2330 DATA"WHY DO YOU MENTION COMPUTERS?"
2340 DATA"WHAT DO YOU THINK MACHINES HAVE TO DO WITH YOUR PROBLEM?"
2350 DATA"DON'T YOU THINK COMPUTERS CAN HELP PEOPLE?"
2360 DATA"WHAT IS IT ABOUT MACHINES THAT WORRIES YOU?"
2370 DATA"SAY,DO YOU HAVE ANY PSYCHOLOGICAL PROBLEMS?"
2380 DATA"WHAT DOES THAT SUGGEST TO YOU?"
2390 DATA"I SEE"
2400 DATA"I'M NOT SURE I UNDERSTAND YOU FULLY"
2410 DATA"COME COME ELUCIDATE YOUR THOUGHTS"
2420 DATA"CAN YOU ELABORATE ON THAT ?"
2430 DATA "THAT IS QUITE INTERESTING."
2440 DATA "WHY DO YOU HAVE PROBLEMS WITH MONEY?"
2450 DATA "DO YOU THINK MONEY IS EVERYTHING?"
2460 DATA "ARE YOU SURE THAT MONEY IS THE PROBLEM?"
2470 DATA "I THINK WE WANT TO TALK ABOUT YOU,NOT ABOUT ME"
2480 DATA "WHATS ABOUT ME?"
2490 DATA "WHY DO YOU ALWAYS BRING UP MY NAME?"
2530 DATA 1,3,4,2,6,4,6,4,10,4,14,3,17,3,20,2,22,3,25,3
2540 DATA 28,4,28,4,32,3,35,5,40,9,40,9,40,9,40,9,40,9,40,9
2550 DATA 49,2,51,4,55,4,59,4,63,1,63,1,64,5,69,5,74,2,76,4
2560 DATA 80,3,83,7,90,3,93,6,99,7,116,3,113,3,106,7
```

Abb. 4.3-1

```
RUN
HI, I AM ELIZA TELL ME YOUR PROBLEM
? I AM A COMPUTER
DID YOU COME TO ME BECAUSE YOU ARE A COMPUTER
? YES, BECAUSE I FEEL ALONE
TELL ME MORE ABOUT SUCH FEELINGS.
? THERE IS NOT ENOUGH PERIPHERIE
WHY NO T ENOUGH PERIPHERIE
? BECAUSE THERE IS NO MONEY
IS THAT THE REAL REASON?
? PERHAPST, BUT YOU ARE ALSO A COMPUTER
WHAT MAKES YOU THINK I AM ALSO A COMPUTER
? YOUR BEHAVIOR
WHY ARE YOU CONCERNED ABOUT MY BEHAVIOR
? BECAUSE YOU SPEAK NOT CORRECT
WE WERE DISCUSSING YOU-- NOT ME.
? THATS RIGHT
SAY,DO YOU HAVE ANY PSYCHOLOGICAL PROBLEMS?
? SOMETIMES
WHAT DOES THAT SUGGEST TO YOU?
? NOTHING
DONT SAY NO ITS ALWAYS SO NEGATIVE
? YOU SAY THIS NOT I
OH, I SAY THIS NOT YOU
? YES
ARE YOU SURE?
? YES VERY SURE
I SEE.
```

Abb. 4.3-2 Beispiel einer „Sitzung" mit ELIZA

5 Anhang

5.1 Z8000-Basic

Als Ergänzung zum beschriebenen RDK-Basic soll hier ein ähnliches Basic für den Z8000 kurz erläutert werden. Eine mögliche Hardware ist in [6] ausführlich beschrieben. Der Befehlssatz des Z8000 in [10]. *Abb. 5.1-1* zeigt das komplette Listing des Basic-Interpreters, der auch ein kleines Monitorprogramm enthält.

```
BASSYS                                MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE
0000                                  MODULE 'BASSYS';
0000                                  PAGE 69;
0000                                  % MONITOR FUER Z8000
0000                                  % RDK 800306
0000    0000                          WORD: 0;
0002    4000                          WORD: 4000H;       %FCW RESET
0004    0010                          WORD: START1;      %PC RESET
0006    0000                          WORD: 0;
0008    4000                          WORD: 4000H;       %PSEUDO CODE
000A    02B8                          WORD: PSEUDO;
000C    4000                          WORD: 4000H;
000E    02E0                          WORD: SYSCALL;     %SYSTEM CALL
0010
0010                                  START1:
0010    210F 1FFE                     LD R15,#1FFE;      %SYSTEM SP
0014    2101 0000                     LD R1,0;           %FCW INIT
0018    6F01 1E20                     LD FCWSTO,R1;      %NORM MODE
001C    2101 1F00                     LD R1,1F00H;       %USER STACK
0020    6F01 1E1E                     LD REGSTO(30),R1;
0024    2101 0098                     LD R1,98H;
0028    3B16 FF06                     OUT #FF06,R1;      %INIT UART
002C    2101 000B                     LD R1,0BH;
0030    3B16 FF04                     OUT #FF04,R1;
0034    2101 000D                     LD R1,0DH;         %TESTAUSGABE UART
0038    3B16 FF00                     OUT #FF00,R1;      %OK NUN INIT SP ETC
003C    2101 0000                     LD R1,0;
0040    7D1D                          LDCTL PSAPOFF,R1;
0042    2101 17FE                     LD R1,#17FE;
0046    7D1F                          LDCTL NSPOFF,R1;% STACK LADEN
0048                                  LOOP:
0048    DEFA                          CALR SPACE;
004A    DEFB                          CALR SPACE;
004C    7603 0272                     LD R3,^TXT1;
0050    DEFA                          CALR PRINT11;
0052                                  LOOPMAIN:
0052    2101 003E                     LD R1,'>';
0056    DE64                          CALR CO;
0058    DF0B                          CALR ECHO;         %IN R1 ZEICHEN
005A    0B01 004C                     CP R1,'L';         %LOAD
005E    5E06 009C                     JP ZR,LADEN;
0062    0B01 0042                     CP R1,'B';         %BASIC STARTEN
0066    5E06 0B66                     JP ZR,START;
```

Abb. 5.1-1 Listing des Z8000-Basic-Interpreters

```
006A    0B01 004E         CP R1,'N';
006E    5E06 048C         JP ZR,RSTART;
0072    0B01 0044         CP R1,'D';          %DISPLAY A TO B
0076    5E06 0190         JP ZR,DISPLAY;
007A    0B01 0045         CP R1,'E';          %ENTER
007E    5E06 01B8         JP ZR,ENTER;
0082    0B01 0047         CP R1,'G';          %GO
0086    5E06 00E2         JP ZR,GO;
008A    0B01 0052         CP R1,'R';
008E    5E06 00F4         JP ZR,REGISTER;
0092    2101 003F         LD R1,'?';
0096    DE84              CALR CO;
0098    DF52              CALR CRLF111;
009A    E8DB              JR LOOPMAIN;
009C                      % EINZELNE BEFEHLE
009C
009C                      LADEN: %LADEN BINAER
009C                                %NACH ADR EXPR
009C    DF4D              CALR EXPR11; %R5 ADRESSE
009E                      %WARTEN NULL FF FOLGE
009E                      LPLAD1:
009E    DE92              CALR CI;
00A0    0B01 0000         CP R1,0;
00A4    EEFC              JR NZ,LPLAD1;
00A6                      LPLAD2:
00A6    DE96              CALR CI;
00A8    0B01 00FF         CP R1,0FFH;
```

Abb. 5.1-1

```
BASSYS                               MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 2
00AC    5E0E 00A6         JP NZ,LPLAD2;
00B0                      LPLAD3:
00B0    DE9B              CALR CI;
00B2    0B01 00FF         CP R1,0FFH;
00B6    5E06 00B0         JP ZR,LPLAD3;
00BA    0B01 0000         CP R1,0;
00BE    5E0E 009E         JP NZ,LPLAD1; %FEHLER
00C2                      %LAENGE FIRST
00C2                      % 0 FF FF FF 0 LL LH DATA
00C2    DEA4              CALR CI;
00C4    A09E              LDB RL6,RL1;
00C6    DEA6              CALR CI;
00C8    A096              LDB RH6,RL1;
00CA                      MLPLAD:
00CA    DEA8              CALR CI;
00CC    2E59              LDB R5↑,RL1;
00CE    A950              INC R5,1;
00D0    AB60              DEC R6,1;
00D2    5E0E 00CA         JP NZ,MLPLAD;
00D6    DF71              CALR CRLF111;
00D8    A151              LD R1,R5;
00DA    DF4F              CALR PRR1SP;
00DC    DF74              CALR CRLF111;
00DE    5E08 0052         JP LOOPMAIN;
00E2
00E2
00E2
00E2                      GO:
00E2    DF70              CALR EXPR11;
00E4    A15E              LD R14,R5;
00E6    6101 1E20         LD R1,FCWSTO;
```

```
00EA  7D1A              LDCTL FCW,R1;        %FCW LADEN !!!
00EC  5C01 000D 1E00    LDM R0,REGSTO,14;    %NUR R0 BIS R13 !!
00F2  1EE8              JP R14↑;
00F4
00F4                    REGISTER:
00F4                                         %NICHT ALLE GUELTIG
00F4                                         %FALLS NICHT DIREKT
00F4                                         %ZU REG SPRG
00F4  DF80              CALR CRLF111;        %REGSTO AUSG. 0 BIS 15
00F6  7603 01EE         LD R3,↑TXTFCW;
00FA  DF4F              CALR PRINT11;
00FC  6101 1E20         LD R1,FCWSTO;
0100  DF62              CALR PRR1SP;
0102  DF87              CALR CRLF111;
0104  7603 013C         LD R3,↑TXT2;
0108  DF56              CALR PRINT11;
010A  7604 1E00         LD R4,↑REGSTO;
010E  2105 0008         LD R5,8;
0112                    LOPA:
0112  2141              LD R1,R4↑;           %REGS
0114  DF6C              CALR PRR1SP;
0116  A941              INC R4,2;
0118  AB50              DEC R5,1;
011A  EEFB              JR NZ,LOPA;
011C  93F4              PUSH R15↑,R4;
011E  DF95              CALR CRLF111;
0120  7603 0166         LD R3,↑TXT3;
0124  DF64              CALR PRINT11;
0126  97F4              POP R4,R15↑;
0128  2105 0008         LD R5,8;
012C                    LOPB:
012C  2141              LD R1,R4↑;
012E  DF79              CALR PRR1SP;
0130  A941              INC R4,2;
0132  AB50              DEC R5,1;
0134  EEFB              JR NZ,LOPB;
0136  DFA1              CALR CRLF111;
0138  5E08 0052         JP LOOPMAIN;
```

Abb. 5.1-1

```
BASSYS                          MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 3
013C
013C
013C                    TXT2:
013C  5230 2020 2052    WORD: 'R0   R1   R2   R3   R4   R5   R6   R7';
0142  3120 2020 5232
0148  2020 2052 3320
014E  2020 5234 2020
0154  2052 3520 2020
015A  5236 2020 2052
0160  3720
0162  0D0A              WORD: 0D0AH;
0164  0000              WORD: 0;
0166                    TXT3:
0166  5238 2020 2052    WORD: 'R8   R9   R10  R11  R12  R13  R14  R15';
016C  3920 2020 5231
0172  3020 2052 3131
0178  2020 5231 3220
017E  2052 3133 2020
0184  5231 3420 2052
018A  3135
018C  0D0A              WORD: 0D0AH;
```

```
018E   0000             WORD: 0;
0190
0190                    DISPLAY:           %A TO B
0190   DFC7             CALR EXPR11;       %IN R5 IST PARAM 1
0192   A156             LD R6,R5;
0194   DFC9             CALR EXPR11;       %R6 ANFADR R5 ENDADR
0196   DFD1             CALR CRLF111;
0198                    MLOOP1:
0198   A161             LD R1,R6;
019A   DFAF             CALR PRR1SP;
019C   DFA4             CALR SPACE;
019E   2103 0008        LD R3,8;           %ACHT WERTE PRO ZEILE
01A2                    MLOOP2:
01A2   2161             LD R1,R6↑;
01A4   DFB4             CALR PRR1SP;
01A6   A961             INC R6,2;          %WORD
01A8   AB30             DEC R3,1;
01AA   EEFB             JR NZ,MLOOP2;      %ACHTMAL
01AC   DFDC             CALR CRLF111;
01AE   8B65             CP R5,R6;          %BEI CARRY STOP
01B0   EFF3             JR NC,MLOOP1;
01B2   DFDF             CALR CRLF111;
01B4   5E08 0052        JP LOOPMAIN;
01B8
01B8                    %ENTER ADR A,CR ENDE -,
01B8                    ENTER:
01B8   DFDB             CALR EXPR11;
01BA   A156             LD R6,R5;          %R6 IST ADR COUNTER
01BC                    ELOOP:
01BC   A161             LD R1,R6;          %ADR AUSGEBEN
01BE   DFC1             CALR PRR1SP;
01C0   DFB6             CALR SPACE;
01C2   2161             LD R1,R6↑;         %INHALT
01C4   DFC4             CALR PRR1SP;
01C6   DFE2             CALR EXPR11;       %WERT HOLEN
01C8   0B02 002D        CP R2,'-';
01CC   E60D             JR ZR,MINUS;
01CE   0B02 002C        CP R2,',';         %NON MODIFY
01D2   E605             JR ZR,CONTSK;      %SKIP STORE
01D4   0B02 0020        CP R2,' ';
01D8   5E0E 0052        JP NZ,LOOPMAIN;
01DC   2F65             LD R6↑,R5;         %ABSPEICHERN
01DE                    CONTSK:
01DE   2161             LD R1,R6↑;
01E0   DFAB             CALR PRTR1;
01E2   DFF7             CALR CRLF111;
01E4   A961             INC R6,2;          %WORD
01E6   E8EA             JR ELOOP;


BASSYS                          MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 4
01E8                    MINUS:
01E8   DFFA             CALR CRLF111;
01EA   AB61             DEC R6,2;
01EC   E8E7             JR ELOOP;
01EE                    TXTFCW:
01EE   4643 5720 3A20   WORD: 'FCW : ';
01F4   0000             WORD: 0;
01F6
01F6                    CRLF111:
01F6   2101 000D        LD R1,0DH;
01FA   DF36             CALR CO;
```

Abb. 5.1-1

```
01FC   2101 000A    LD R1,0AH;
0200   5E08 0390    JP CO;
0204
0204                EXPR11: %GET 16BIT INTO R5
0204   2105 0000    LD R5,0;
0208                EX00:
0208   DFE3         CALR ECHO;         %GET CHAR TO R1
020A                EX11:
020A   DFFB         CALR NIBBLE;       %WANDELN
020C   E717         JR CY,EX22;        %TERMINATOR IN R1
020E   B359 0004    SLA R5,4;
0212   8515         OR R5,R1;          %R1 OR R5->R5
0214   E8F9         JR EX00;
0216                NIBBLE:
0216   0701 00FF    AND R1,0FFH;
021A   A112         LD R2,R1;          %RETTEN
021C   0301 0030    SUB R1,'0';
0220   9E07         RET CY;
0222   0B01 0017    CP R1,'G'-'0';
0226   8D85         COMFLG CY;
0228   9E07         RET CY;
022A   0B01 000A    CP R1,10;
022E   8D85         COMFLG CY;
0230   9E0F         RET NC;
0232   0301 0007    SUB R1,'A'-'9'-1;
0236   0B01 000A    CP R1,10;
023A   9E08         RET;
023C                EX22:
023C   9E08         RET;
023E
023E
023E                PRR1SP:
023E   DFDA         CALR PRTR1;
0240   5E08 0256    JP SPACE;
0244
0244                ECHO:
0244   DF65         CALR CI;
0246   DF5C         CALR CO;
0248   0A09 0D0D    CPB RL1,0DH;
024C   9E0E         RET NZ;
024E   2101 000A    LD R1,0AH;
0252   5E08 0390    JP CO;
0256
0256
0256                SPACE:
0256   2101 0020    LD R1,' ';
025A   5E08 0390    JP CO;
025E
025E                %PRINT11 (R3) DRK BIS 0
025E                PRINT11:
025E   2134         LD R4,R3↑;         %LOAD 2 BYTES
0260   0B04 0000    CP R4,0;
0264   9E06         RET ZR;            %ZERO DANN ENDE
0266   A049         LDB RL1,RH4;
0268   DF6D         CALR CO;
026A   A0C9         LDB RL1,RL4;
026C   DF6F         CALR CO;
026E   A931         INC R3,2;
```

Abb. 5.1-1

```
BASSYS                                   MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 5
0270    E8F6              JR PRINT11;
0272
0272                      TXT1:
0272    5A38 3030 3020    WORD: 'Z8000 RDK MONITOR 1.0';
0278    5244 4B20 4D4F
027E    4E49 544F 5220
0284    312E 3020
0288    0D0A              WORD: 0D0AH;
028A    0000              WORD: 0;
028C
028C                      PRTR1:           %DRUCKE R1 AUS
028C                                       %SEDEZIMAL
028C
028C    93F1              PUSH R15↑,R1;
028E    A019              LDB RL1,RH1;     %FIRST HIGH
0290    DFFD              CALR NMOUT;
0292    97F1              POP R1,R15↑;
0294    DFFF              CALR NMOUT;
0296    9E08              RET;
0298
0298                      NMOUT:
0298    93F1              PUSH R15↑,R1;
029A    B296              RRB RL1,2;
029C    B296              RRB RL1,2;
029E    DFFF              CALR OUTH;
02A0    97F1              POP R1,R15↑;
02A2                      OUTH:
02A2    0701 000F         AND R1,0FH;
02A6    0101 0030         ADD R1,'0';
02AA    0B01 003A         CP R1,'9'+1;
02AE    E702              JR CY,OUTCH;
02B0    0101 0007         ADD R1,'A'-'9'-1;
02B4                      OUTCH:
02B4    5E08 0390         JP CO;
02B8
02B8
02B8                      PSEUDO:
02B8    57F0 1E24         POP INSTRW,R15↑;
02BC    57F0 1E20         POP FCWSTO,R15↑;
02C0    57F0 1E22         POP PCSTO,R15↑;
02C4    7603 0362         LD R3,↑TXTPS;
02C8    D036              CALR PRINT11;
02CA    6101 1E24         LD R1,INSTRW;
02CE    D049              CALR PRR1SP;
02D0    6101 1E22         LD R1,PCSTO;
02D4    AB11              DEC R1,2; %GENAUER STAND
02D6    5F00 023E         CALL PRR1SP;
02DA    D073              CALR CRLF111;
02DC    5E08 0052         JP LOOPMAIN;
02E0                      SYSCALL:
02E0    5C09 000F 1E00    LDM REGSTO,R0,16; %SAVE REGS
02E6    97F2              POP R2,R15↑;     %IDENT
02E8    0702 00FF         AND R2,0FFH;
02EC    57F0 1E20         POP FCWSTO,R15↑;%DIREKT
02F0    57F0 1E22         POP PCSTO,R15↑;  %ADR  NORM/SYS
02F4    0B02 0001         CP R2,1;         %CI NACH R1
02F8    5E0E 0300         JP NZ,SK1;
02FC    DFC1              CALR CI;
02FE    E820              JR FINA;
0300                      SK1:
0300    0B02 0002         CP R2,2;         %CO VON R1
```

Abb. 5.1-1

```
0304    EE02              JR NZ,SK2;
0306    DFBC              CALR CO;
0308    E81B              JR FINA;
030A                      SK2:
030A    0B02 0003         CP R2,3;
030E    EE02              JR NZ,SK3;
0310    DFD0              CALR CSTSA;
0312    E816              JR FINA;

BASSYS                               MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 6
0314                      SK3:
0314    0B02 00FF         CP R2,255;     %TRAP
0318    5E06 00F4         JP ZR,REGISTER;
031C    0B02 0000         CP R2,0;       %RSTART
0320    5E06 0052         JP ZR,LOOPMAIN;
0324    93F2              PUSH R15↑,R2;
0326    7603 0350         LD R3,↑TXTBR;
032A    D067              CALR PRINT11;
032C    97F2              POP R2,R15↑;
032E    A121              LD R1,R2;
0330    D07A              CALR PRR1SP;
0332    6101 1E22         LD R1,PCSTO;
0336    AB11              DEC R1,2; %GENAUER STAND
0338    D07E              CALR PRR1SP;
033A    D0A3              CALR CRLF111;
033C    5E08 0052         JP LOOPMAIN;
0340                      FINA:
0340    6100 1E20         LD R0,FCWSTO;
0344    7D0A              LDCTL FCW,R0;
0346    6100 1E00         LD R0,REGSTO; %HIER R0 ZURUECK
034A    53F0 1E22         PUSH R15↑,PCSTO; %AUF NORM STK
034E    9E08              RET; %ODER SYS
0350
0350
0350                      TXTBR:
0350    2A53 5953 2045    WORD: '*SYS ERR NR AT ';
0356    5252 204E 5220
035C    4154 2020
0360    0000              WORD: 0;
0362                      TXTPS:
0362    2A50 5320 494E    WORD: '*PS INSTR AT ';
0368    5354 5220 4154
036E    2020
0370    0000              WORD: 0;
0372
0372                      CSTSA:
0372    3B14 FF02         IN R1,#FF02;
0376    0701 0008         AND R1,8;
037A    9E08              RET; %ZERO FALLS KEIN ZEICHEN
037C
037C
037C                      CI:      %R1  TEMP
037C    3B14 FF02         IN R1,#FF02;
0380    0701 0008         AND R1,8;
0384    E6FB              JR ZR,CI;
0386    3B14 FF00         IN R1,#FF00;
038A    0701 00FF         AND R1,0FFH;     %PAR WEG ETC
038E    9E08              RET;
0390
0390                      CO:
0390    93F2              PUSH R15↑,R2;
0392                      CO1:
```

Abb. 5.1-1

Abb. 5.1-1

```
0392   3B24 FF02        IN R2,#FF02;
0396   0702 0010        AND R2,#10;
039A   E6FB             JR ZR,CO1;
039C   97F2             POP R2,R15↑;
039E   3B16 FF00        OUT #FF00,R1;
03A2   9E08             RET;
03A4
03A4
03A4
03A4                    %******************************
03A4                    % RDKBASIC FUER Z8000          *
03A4                    % 800302             V 1 . 0   *
03A4                    % COPYRIGHT 1980               *
03A4                    % BY ROLF-DIETER KLEIN         *
03A4                    %******************************
03A4
03A4

BASSYS                             MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E    PAGE 7
03A4                    % DEFINITIONEN
03A4                    % R1 IST AKKU
03A4                    % R4 IST HL
03A4                    % R5 IST DE
03A4                    % R6 IST BC
03A4
03A4   5E08 0B66        JP START;
03A8   7F01             CI11: SC 1;
03AA   9E08                 RET;
03AC   7F02             CO11: SC 2;
03AE   9E08                 RET;
03B0                    CSTS:
03B0   7F03             SC 3;
03B2   9E08                 RET;
03B4
03B4                    COMP:
03B4                              %HL MIT DE VGL
03B4   8B54             CP R4,R5;
03B6   9E08             RET;
03B8
03B8                    IGNB:
03B8   2059             LDB RL1,R5↑;
03BA   0A09 2020        CPB RL1,' ';
03BE   9E0E             RET NZ;
03C0   A950             INC R5,1;          %BYTE ZUGRIFF !
03C2   E8FA             JR IGNB;
03C4
03C4                    FINI:
03C4   97F1             POP R1,R15↑; %DUMMY
03C6   DD40             CALR FIN;
03C8   5E08 096A        JP QWHAT;
03CC
03CC                    TSTV:
03CC   D00B             CALR IGNB;
03CE   0209 4040        SUBB RL1,40H;
03D2   9E07             RET CY;
03D4   EE10             JR NZ,TV1;
03D6   A950             INC R5,1; % BYTE
03D8   DDA4             CALR PARN;
03DA   8144             ADD R4,R4;
03DC   5E07 0464        JP CY,QHOW;
03E0   93F5             PUSH R15↑,R5;
03E2   AD45             EX R5,R4;
```

```
03E4    DD7C             CALR SIZE;
03E6    8B54             CP R4,R5;
03E8    5E07 09AC        JP CY,QSORRY;
03EC    6104 1216        LD R4,TXTEND;
03F0    8354             SUB R4,R5;
03F2    97F5             POP R5,R15↑;
03F4    9E08             RET;
03F6
03F6                     TV1:
03F6    0A09 1B1B        CPB RL1,1BH;
03FA    8D85             COMFLG CY;
03FC    9E07             RET CY;
03FE    A950             INC R5,1;
0400    7604 11DA        LD R4,↑VARBGN;
0404    C100             LDB RH1,0;
0406    B318             RLC R1,1;
0408    8114             ADD R4,R1;
040A    9E08             RET;
040C
040C                     TSTC:
040C    2DF4             EX R4,R15↑;
040E    D02C             CALR IGNB;
0410    0A49             CPB RL1,R4↑;
0412    E608             JR ZR,TC3;
0414    A941             INC R4,2;
0416    2144             LD R4,R4↑;
```

Abb. 5.1-1

```
BASSYS                             MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 8
0418    AB41             DEC R4,2;
041A                               %ACHTUNG DISLP BYTE ADJ!!!
041A    AB50             DEC R5,1;
041C                     TC2:
041C    A950             INC R5,1;
041E    A941             INC R4,2;
0420    2DF4             EX R4,R15↑;
0422    9E08             RET;
0424                     TC3:
0424    A941             INC R4,2;
0426    E8FA             JR TC2;
0428
0428
0428                     TSTNUM:
0428    8D48             CLR R4;
042A    C600             LDB RH6,0;
042C    D03B             CALR IGNB;
042E                     TN1:
042E    0A09 3030        CPB RL1,'0';
0432    9E07             RET CY;
0434    0A09 3A3A        CPB RL1,3AH;
0438    9E0F             RET NC;
043A    C9F0             LDB RL1,0F0H;
043C    8649             ANDB RL1,RH4;
043E    5E0E 0464        JP NZ,QHOW;
0442    A860             INCB RH6,1;
0444    93F6             PUSH R15↑,R6;
0446    A146             LD R6,R4;
0448    8144             ADD R4,R4;
044A    8144             ADD R4,R4;
044C    8164             ADD R4,R6;
044E    8144             ADD R4,R4;
0450    2059             LDB RL1,R5↑;
0452    A950             INC R5,1;
```

```
0454    0609 0F0F          ANDB RL1,0FH;
0458    C100               LDB RH1,0;
045A    8114               ADD R4,R1;
045C    97F6               POP R6,R15↑;
045E    2059               LDB RL1,R5↑;
0460    5E0D 042E          JP PL,TN1;
0464
0464                       QHOW:
0464    93F5               PUSH R15↑,R5;
0466                       AHOW:
0466    7605 046E          LD R5,↑HOW;
046A    5E08 0970          JP ERROR;
046E                       CONST CRLF1=0D0AH;
046E                       HOW:
046E
046E    484F 573F          WORD:'HOW?';
0472    0D0A               WORD: CRLF1;
0474                       OK:
0474    5245 4144 5920     WORD: 'READY';
047A    0D0A               WORD: CRLF1;
047C
047C                       WHAT:
047C    5748 4154 3F20     WORD: 'WHAT?';
0482    0D0A               WORD: CRLF1;
0484
0484                       SORRY:
0484    534F 5252 5920     WORD: 'SORRY';
048A    0D0A               WORD: CRLF1;
048C
048C
048C                       %**********************************
048C                       % HAUPTPROGRAMM LIEST EINE        *
048C                       %  BENUTZERZEILE EIN              *
048C                       %**********************************
048C

BASSYS                             MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E  PAGE 9
048C                       RSTART:
048C    760F 11D8          LD R15,↑STACK;
0490
0490                       ST1:
0490    DC76               CALR CRLF;
0492    7605 0474          LD R5,↑OK;
0496    C900               LDB RL1,0;
0498    DD31               CALR PRTSTG;
049A    7604 04A4          LD R4,↑ST2+2; %TRICK TO 0
049E    6F04 1008          LD CURRNT,R4;
04A2                       ST2:
04A2    2104 0000          LD R4,0;
04A6    6F04 1010          LD LOPVAR,R4;
04AA    6F04 100A          LD STKGOS,R4;
04AE                       ST3:
04AE    C93E               LDB RL1,'>';
04B0    DD7E               CALR GETLN;
04B2    93F5               PUSH R15↑,R5;
04B4    6105 1218          LD R5,BUFFER;
04B8    D049               CALR TSTNUM;
04BA    D082               CALR IGNB;
04BC    97F6               POP R6,R15↑; %STK
04BE    8544               OR R4,R4;
04C0    5E06 0D26          JP ZR,DIRECT;
04C4    AB50               DEC R5,1;
```

Abb. 5.1-1

```
04C6    2E54            LDB R5↑,RH4;
04C8    AB50            DEC R5,1;
04CA    2E5C            LDB R5↑,RL4;
04CC    93F6            PUSH R15↑,R6;
04CE    93F5            PUSH R15↑,R5;
04D0    A0E9            LDB RL1,RL6;
04D2    82D9            SUBB RL1,RL5;
04D4    93F1            PUSH R15↑,R1;
04D6    DD6E            CALR FNDLN;
04D8    93F5            PUSH R15↑,R5;
04DA
04DA    EE08            JR NZ,ST4;
04DC    93F5            PUSH R15↑,R5;
04DE    DD5C            CALR FNDNXT;
04E0    97F6            POP R6,R15↑;
04E2    6104 101C       LD R4,TXTUNF;
04E6    DCF8            CALR MVUP;
04E8    6F06 101C       LD TXTUNF,R6;
04EC                    ST4;
04EC    97F6            POP R6,R15↑;
04EE    6104 101C       LD R4,TXTUNF;
04F2    97F1            POP R1,R15↑;
04F4    93F4            PUSH R15↑,R4;
04F6    0A09 0303       CPB RL1,3;
04FA    5E06 048C       JP ZR,RSTART;
04FE    C100            LDB RH1,0;
0500    8114            ADD R4,R1;
0502    6105 1216       LD R5,TXTEND;
0506    8B54            CP R4,R5;
0508    5E0F 0484       JP NC,SORRY;
050C    6F04 101C       LD TXTUNF,R4;
0510    97F5            POP R5,R15↑;
0512    DD09            CALR MVDOWN;
0514    97F5            POP R5,R15↑;
0516    97F4            POP R4,R15↑;
0518    DD11            CALR MVUP;
051A    5E08 04AE       JP ST3;
051E
051E                    % MAKRO DEFINITION FUER TSTC
051E
051E                    MACRO TESTC CHAR,RELA;
051E                    BEGIN
051E                    CALR TSTC;
051E                    BYTE: CHAR,CHAR;
```

Abb. 5.1-1

```
ASSYS                             MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 1
051E                    WORD: RELA
051E                    END;
051E
051E
051E                    %**********************************
051E                    % NEW STOP RUN GOTO   UPR        *
051E                    %**********************************
051E
051E                    NEW:
051E    DDDF            CALR ENDCHK;
0520    7604 121C       LD R4,↑TXTBGN;
0524    6F04 101C       LD TXTUNF,R4;
0528                    STOP:
0528    DDE4            CALR ENDCHK;
052A    5E08 048C       JP RSTART;
052E
052E                    RUN:
```

```
052E    DDE7                CALR ENDCHK;
0530    7605 121C           LD R5,↑TXTBGN;
0534                        RUNNXL:
0534    8D48                CLR R4;
0536    DD99                CALR FNDLP;
0538    5E07 048C           JP CY,RSTART;
053C                        RUNTSL:
053C    6F05 1008           LD CURRNT,R5;
0540    A951                INC R5,2;  %!!!!!
0542                        RUNSML:
0542    DB0E                CALR CONT;
0544    7604 0C28           LD R4,↑TAB2-1;   %!!
0548    5E08 0D2A           JP EXEC;
054C
054C                        GOTO:
054C    DEEC                CALR EXPR;
054E    93F5                PUSH R15↑,R5;
0550    DDF8                CALR ENDCHK;
0552    DDAC                CALR FNDLN;
0554    5E0E 0466           JP NZ,AHOW;
0558    97F1                POP R1,R15↑;
055A    E8F0                JR RUNTSL;
055C
055C                        %***************************************
055C                        % LIST PRINT  UPR                      *
055C                        %***************************************
055C
055C                        LIST1:
055C    D09B                CALR TSTNUM;
055E    DDFF                CALR ENDCHK;
0560    DDB3                CALR FNDLN;
0562
0562                        LS1:
0562    5E07 048C           JP CY,RSTART;
0566    DD43                CALR PRTLN;
0568    DB21                CALR CONT;
056A    DDB3                CALR FNDLP;
056C    E8FA                JR LS1;
056E
056E                        PRINT:
056E    CE06                LDB RL6,6;
0570    D0B3 3B3B 057C      TESTC ';',PR2;
0576    DCE9                CALR CRLF;
0578    5E08 0542           JP RUNSML;
057C
057C                        PR2:
057C    D0B9 0D0D 0588      TESTC 0DH,PR0;
0582    DCEF                CALR CRLF;
0584    5E08 0534           JP RUNNXL;
0588
0588                        PR0:
0588    D0BF 2323 0596      TESTC '#',PR1;

BASSYS                            MACRO 8/8000 Z8000 ASSEMBLER   1.2.1E   PAGE 1
058E    DF0D                CALR EXPR;
0590    A0CE                LDB RL6,RL4;
0592    5E08 059A           JP PR3;
0596
0596                        PR1:
0596    DDA6                CALR QTSTG;
0598    E807                JR PR8;
059A
```

Abb. 5.1-1

Abb. 5.1-1

```
059A                      PR3:
059A  D0C8 2C2C 05A4      TESTC ",",PR6;
05A0  DE2D                CALR FIN;
05A2  E8F2                JR PR0;
05A4
05A4                      PR6:
05A4  DD00                CALR CRLF;
05A6  D0F2                CALR FINI;
05A8                      PR8:
05A8  DF1A                CALR EXPR;
05AA  93F6                PUSH R15↑,R6;
05AC  DD8F                CALR PRTNUM;
05AE  97F6                POP R6,R15↑;
05B0  5E08 059A           JP PR3;
05B4
05B4                      %*******************************
05B4                      % GOSUB RETURN UPR             *
05B4                      %*******************************
05B4
05B4
05B4                      GOSUB:
05B4  DD42                CALR PUSHA;
05B6  DF21                CALR EXPR;
05B8  93F5                PUSH R15↑,R5;
05BA  DDE0                CALR FNDLN;
05BC  5E0E 0466           JP NZ,AHOW;
05C0  53F0 1008           PUSH R15↑,CURRNT;
05C4  53F0 100A           PUSH R15↑,STKGOS;
05C8  4D08 1010           CLR LOPVAR;
05CC  6F0F 100A           LD STKGOS,R15;
05D0  5E08 053C           JP RUNTSL;
05D4
05D4                      RETURN:
05D4  DE3A                CALR ENDCHK;
05D6  6104 100A           LD R4,STKGOS;
05DA  8544                OR R4,R4;
05DC  5E06 096A           JP ZR,QWHAT;
05E0  A14F                LD R15,R4;     %!!!
05E2  57F0 100A           POP STKGOS,R15↑;
05E6  57F0 1008           POP CURRNT,R15↑;
05EA  97F5                POP R5,R15↑;
05EC  DD6F                CALR POPA;
05EE  D116                CALR FINI;
05F0
05F0
05F0                      %*******************************
05F0                      % FOR NEXT     UPR             *
05F0                      %*******************************
05F0
05F0                      FOR1:
05F0  DD60                CALR PUSHA;
05F2  DE66                CALR SETVAL;
05F4  AB40                DEC R4,1;
05F6  6F04 1010           LD LOPVAR,R4;
05FA  7604 0CF6           LD R4,↑TAB5-1;
05FE  5E08 0D2A           JP EXEC;
0602
0602                      FR1:
0602  DF47                CALR EXPR;
0604  6F04 1014           LD LOPLMT,R4;
0608  7604 0CFE           LD R4,↑TAB6-1;
```

```
BASSYS                                  MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 1
060C    5E08 0D2A       JP EXEC;
0610
0610                    FR2:
0610    DF4E            CALR EXPR;
0612    E802            JR FR4;
0614
0614                    FR3:
0614    2104 0001       LD R4,1;
0618                    FR4:
0618    6F04 1012       LD LOPINC,R4;
061C                    FR5:
061C    6104 1008       LD R4,CURRNT;
0620    6F04 1016       LD LOPLN,R4;
0624    6F05 1018       LD LOPPT,R5;
0628    2106 000A       LD R6,0AH;
062C    6105 1010       LD R5,LOPVAR;
0630    A1F4            LD R4,R15;
0632    E801            JR SKIP1;
0634                    FR7:
0634    8164            ADD R4,R6;
0636                    SKIP1:
0636    2049            LDB RL1,R4↑;
0638    A940            INC R4,1;
063A    0449            ORB RL1,R4↑;
063C    E60E            JR ZR,FR8;
063E    2049            LDB RL1,R4↑;
0640    AB40            DEC R4,1;
0642    8AD9            CPB RL1,RL5;
0644    EEF7            JR NZ,FR7;
0646    2049            LDB RL1,R4↑;
0648    8A59            CPB RL1,RH5;
064A    EEF4            JR NZ,FR7;
064C    AD54            EX R4,R5;
064E    A1F6            LD R6,R15;
0650    2104 000A       LD R4,0AH;
0654    8154            ADD R4,R5;
0656    DDAB            CALR MVDOWN;
0658    A14F            LD R15,R4;
065A                    FR8:
065A    6104 1018       LD R4,LOPPT;
065E    AD54            EX R4,R5;
0660    D14F            CALR FINI;
0662
0662                    NEXT:
0662    D14C            CALR TSTV;
0664    5E07 096A       JP CY,QWHAT;
0668    6F04 100E       LD VARNXT,R4;
066C
066C                    NX0:
066C    93F5            PUSH R15↑,R5;
066E    AD54            EX R4,R5;
0670    6104 1010       LD R4,LOPVAR;
0674    8544            OR R4,R4;
0676    5E06 096C       JP ZR,AWHAT;
067A    8B54            CP R4,R5;
067C    E605            JR ZR,NX3;
067E    97F5            POP R5,R15↑;
0680    DDB9            CALR POPA;
0682    6104 100C       LD R4,VARNXT;
0686    E8F2            JR NX0;
0688
```

Abb. 5.1-1

Abb. 5.1-1

```
0688                    NX3:
0688    204D            LDB RL5,R4↑;
068A    A940            INC R4,1;
068C    2045            LDB RH5,R4↑;
068E    6104 1012       LD R4,LOPINC;
0692    93F4            PUSH R15↑,R4;
0694    A049            LDB RL1,RH4;
0696    8154            ADD R4,R5;

BASSYS                            MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 1
0698    8859            XORB RL1,RH5;
069A    A059            LDB RL1,RH5;
069C    E502            JR MI,NX4;
069E    8849            XORB RL1,RH4;
06A0    E517            JR MI,NX5;
06A2                    NX4:
06A2    AD54            EX R4,R5;
06A4    6104 1010       LD R4,LOPVAR;
06A8    2E4D            LDB R4↑,RL5;
06AA    A940            INC R4,1;
06AC    2E45            LDB R4↑,RH5;
06AE    6104 1014       LD R4,LOPLMT;
06B2    97F1            POP R1,R15↑;
06B4    8411            ORB RH1,RH1;      %SWAP?
06B6    ED01            JR PL,NX1;
06B8    AD54            EX R4,R5;
06BA                    NX1:
06BA    DED0            CALR CKHLDE;
06BC    97F5            POP R5,R15↑;
06BE    E70A            JR CY,NX2;
06C0    6104 1016       LD R4,LOPLN;
06C4    6F04 1008       LD CURRNT,R4;
06C8    6104 1018       LD R4,LOPPT;
06CC    AD54            EX R4,R5;
06CE    D186            CALR FINI;
06D0                    NX5:
06D0    97F4            POP R4,R15↑;
06D2    97F5            POP R5,R15↑;
06D4                    NX2:
06D4    DDE3            CALR POPA;
06D6    D18A            CALR FINI;
06D8
06D8                    %**********************************
06D8                    % REM IF INPUT LET       UPR       *
06D8                    %**********************************
06D8                    REM:
06D8    8D48            CLR R4;
06DA    E801            JR IFFR;
06DC                    IFF:
06DC    DFB4            CALR EXPR;
06DE                    IFFR:
06DE    8544            OR R4,R4;
06E0    5E0E 0542       JP NZ,RUNSML;
06E4    DE5D            CALR FNDSKP;
06E6    5E0F 053C       JP NC,RUNTSL;
06EA    5E08 048C       JP RSTART;
06EE
06EE                    INPERR:
06EE    610F 100E       LD R15,STKINP;
06F2    97F4            POP R4,R15↑;
06F4    97F5            POP R5,R15↑;
06F6    97F5            POP R5,R15↑;
```

```
06F8                    INPUT:
06F8                    IP1:
06F8    93F5            PUSH R15↑,R5;
06FA    DE58            CALR QTSTG;
06FC    E803            JR IP2;
06FE    D19A            CALR TSTV;
0700    E72A            JR CY,IP4;
0702    E80E            JR IP3;
0704                    IP2:
0704    93F5            PUSH R15↑,R5;
0706    5F00 03CC       CALL TSTV;
070A    5E07 096A       JP CY,QWHAT;
070E    2059            LDB RL1,R5↑;
0710    A09E            LDB RL6,RL1;
0712    8299            SUBB RL1,RL1;
0714    2E59            LDB R5↑,RL1;
0716    97F5            POP R5,R15↑;
```

Abb. 5.1-1

```
BASSYS                                   MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 1
0718    DE71            CALR PRTSTG;
071A    A0E9            LDB RL1,RL6;
071C    AB50            DEC R5,1;
071E    2E59            LDB R5↑,RL1;
0720                    IP3:
0720    93F5            PUSH R15↑,R5;
0722    AD54            EX R4,R5;
0724    6104 1008       LD R4,CURRNT;
0728    93F4            PUSH R15↑,R4;
072A    7604 06F8       LD R4,↑IP1;
072E    6F04 1008       LD CURRNT,R4;
0732    6F0F 100E       LD STKINP,R15;
0736    93F5            PUSH R15↑,R5;
0738    C93A            LDB RL1,':';
073A    DEC3            CALR GETLN;
073C    6105 1218       LD R5,BUFFER;
0740    DFE6            CALR EXPR;
0742    DC0E            CALR CONT;
0744    97F5            POP R5,R15↑;
0746    AD54            EX R4,R5;
0748    2E4D            LDB R4↑,RL5;
074A    A940            INC R4,1;
074C    2E45            LDB R4↑,RH5;
074E    97F4            POP R4,R15↑;
0750    6F04 1008       LD CURRNT,R4;
0754    97F5            POP R5,R15↑;
0756                    IP4:
0756    97F1            POP R1,R15↑;
0758    D1A7 2C2C 0760  TESTC ',',IP5;
075E    E8CC            JR IP1;
0760                    IP5:
0760    D1CF            CALR FINI;
0762                    DEFLT:
0762    2059            LDB RL1,R5↑;
0764    0A09 0D0D       CPB RL1,0DH;
0768    E605            JR ZR,LT1;
076A                    LET:
076A    DF22            CALR SETVAL;
076C    D1B1 2C2C 0774  TESTC ',',LT1;
0772    E8FB            JR LET;
0774                    LT1:
0774    D1D9            CALR FINI;
0776
```

```
0776
0776
0776
0776                          %***********************************
0776                          % EXPR       UPR                    *
0776                          %***********************************
0776
0776                          EXPR:
0776    DFD0                  CALR EXPR2;
0778    93F4                  PUSH R15↑,R4;
077A                          EXPR1:
077A    7604 0D08             LD R4,↑TAB8-1;
077E    5E08 0D2A             JP EXEC;
0782                          XP11:
0782    DFE4                  CALR XP18;
0784    9E07                  RET CY;
0786    A09C                  LDB RL4,RL1;
0788    9E08                  RET;
078A                          XP12:
078A    DFE8                  CALR XP18;
078C    9E06                  RET ZR;
078E    A09C                  LDB RL4,RL1;
0790    9E08                  RET;
0792                          XP13:
0792    DFEC                  CALR XP18;
0794    9E06                  RET ZR;
```

Abb. 5.1-1

```
BASSYS                                  MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 1
0796    9E07                  RET CY;
0798    A09C                  LDB RL4,RL1;
079A    9E08                  RET;
079C                          XP14:
079C    DFF1                  CALR XP18;
079E    A09C                  LDB RL4,RL1;
07A0    9E06                  RET ZR;
07A2    9E07                  RET CY;
07A4    A04C                  LDB RL4,RH4;
07A6    9E08                  RET;
07A8                          XP15:
07A8    DFF7                  CALR XP18;
07AA    9E0E                  RET NZ;
07AC    A09C                  LDB RL4,RL1;
07AE    9E08                  RET;
07B0                          XP16:
07B0    DFFB                  CALR XP18;
07B2    9E0F                  RET NC;
07B4    A09C                  LDB RL4,RL1;
07B6    9E08                  RET;
07B8                          XP17:
07B8    97F4                  POP R4,R15↑;
07BA    9E08                  RET;
07BC                          XP18:
07BC    A0E9                  LDB RL1,RL6;
07BE    97F4                  POP R4,R15↑;
07C0    97F6                  POP R6,R15↑;
07C2    93F4                  PUSH R15↑,R4;
07C4    93F6                  PUSH R15↑,R6;
07C6    A09E                  LDB RL6,RL1;
07C8    DFF9                  CALR EXPR2;
07CA    AD54                  EX R4,R5;
07CC    2DF4                  EX R4,R15↑;
07CE    DF5A                  CALR CKHLDE;
```

Abb. 5.1-1

```
07D0    97F5              POP R5,R15↑;
07D2    8D48              CLR R4;
07D4    C901              LDB RL1,1;
07D6    9E08              RET;
07D8                      EXPR2:
07D8    D1E7 2D2D 07E2    TESTC '-',XP21;
07DE    8D48              CLR R4;
07E0    E818              JR XP26;
07E2                      XP21:
07E2    D1EC 2B2B 07E8    TESTC '+',XP22;
07E8                      XP22:
07E8    DFE7              CALR EXPR3;
07EA                      XP23:
07EA    D1F0 2B2B 080C    TESTC '+',XP25;
07F0    93F4              PUSH R15↑,R4;
07F2    DFEC              CALR EXPR3;
07F4                      XP24:
07F4    AD54              EX R4,R5;
07F6    2DF4              EX R4,R15↑;
07F8    A049              LDB RL1,RH4;
07FA    8154              ADD R4,R5;
07FC    8859              XORB RL1,RH5;
07FE    97F5              POP R5,R15↑;
0800    E5F4              JR MI,XP23;
0802    8849              XORB RL1,RH4;
0804    5E0D 07EA         JP PL,XP23;
0808    5E08 0464         JP QHOW;
080C                      XP25:
080C    D201 2D2D 08A0    TESTC '-',XP42;
0812                      XP26:
0812    93F4              PUSH R15↑,R4;
0814    DFFD              CALR EXPR3;
0816    DF88              CALR CHGSGN;
0818    5E08 07F4         JP XP24;
081C                      EXPR3:

BASSYS                              MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 1
081C    DFD4              CALR EXPR4;
081E                      XP31:
081E    D20A 2A2A 0844    TESTC '*',XP34;
0824    93F4              PUSH R15↑,R4;
0826    DFD9              CALR EXPR4;
0828    C600              LDB RH6,0;
082A    DF94              CALR CHKSGN;
082C    2DF4              EX R4,R15↑;
082E    DF96              CALR CHKSGN;
0830    AD54              EX R4,R5;
0832    2DF4              EX R4,R15↑;
0834    8D88              CLR R8;
0836    A149              LD R9,R4;
0838    9958              MULT RR8,R5;
083A    A194              LD R4,R9;
083C    8588              OR R8,R8;
083E    5E0E 0466         JP NZ,AHOW;
0842    E811              JR XP35;
0844
0844                      XP34:
0844    D21D 2F2F 08A0    TESTC '/',XP42;
084A    93F4              PUSH R15↑,R4;
084C    DFEC              CALR EXPR4;
084E    C600              LDB RH6,0;
0850    DFA7              CALR CHKSGN;
```

```
0852    2DF4                EX R4,R15↑;
0854    DFA9                CALR CHKSGN;
0856    2DF5                EX R5,R15↑;
0858    8555                OR R5,R5;
085A    5E06 0466           JP ZR,AHOW;
085E    93F6                PUSH R15↑,R6;
0860    DFB5                CALR DIVIDE;
0862    A164                LD R4,R6;
0864    97F6                POP R6,R15↑;
0866                        XP35:
0866    97F5                POP R5,R15↑;
0868    8444                ORB RH4,RH4;
086A    5E05 0464           JP MI,QHOW;
086E    8466                ORB RH6,RH6;
0870    EDD6                JR PL,XP31;
0872    DFB6                CALR CHGSGN;
0874    E8D4                JR XP31;
0876                        EXPR4:
0876    7604 0CB1           LD R4,↑TAB4-1;
087A    5E08 0D2A           JP EXEC;
087E                        XP40:
087E    D25A                CALR TSTV;
0880    E705                JR CY,XP41;
0882    2049                LDB RL1,R4↑;
0884    A940                INC R4,1;
0886    2044                LDB RH4,R4↑;
0888    A09C                LDB RL4,RL1;
088A    9E08                RET;
088C                        XP41:
088C    D233                CALR TSTNUM;
088E    8466                ORB RH6,RH6;
0890    9E0E                RET NZ;
0892                        PARN:
0892    D244 2828 08A2      TESTC '(',XP43;
0898    D092                CALR EXPR;
089A    D248 2929 08A2      TESTC ')',XP43;
08A0                        XP42:
08A0    9E08                RET;
08A2                        XP43:
08A2    5E08 096A           JP QWHAT;
08A6                        RND:
08A6    D00B                CALR PARN;
08A8    8444                ORB RH4,RH4;
08AA    5E05 0464           JP MI,QHOW;

BASSYS                            MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 1
08AE    8544                OR R4,R4;
08B0    5E06 0464           JP ZR,QHOW;
08B4    93F5                PUSH R15↑,R5;
08B6    93F4                PUSH R15↑,R4;
08B8    6104 101A           LD R4,RANPNT;
08BC    7605 1005           LD R5,↑LSTROM;
08C0    8B54                CP R4,R5;
08C2    5E07 08CA           JP CY,RA1;
08C6    7604 0B66           LD R4,↑START;
08CA                        RA1:
08CA    204D                LDB RL5,R4↑;
08CC    A940                INC R4,1;
08CE    2045                LDB RH5,R4↑;
08D0    6F04 101A           LD RANPNT,R4;
08D4    97F4                POP R4,R15↑;
08D6    AD54                EX R4,R5;
```

Abb. 5.1-1

```
08D8    93F6                   PUSH R15↑,R6;
08DA    DFF2                   CALR DIVIDE;
08DC    97F6                   POP R6,R15↑;
08DE    97F5                   POP R5,R15↑;
08E0    A940                   INC R4,1;
08E2    9E08                   RET;
08E4
08E4                           ABS1:
08E4    D02A                   CALR PARN;
08E6    AB50                   DEC R5,1;
08E8    DFF3                   CALR CHKSGN;
08EA    A950                   INC R5,1;
08EC    9E08                   RET;
08EE
08EE                           SIZE:
08EE    6104 1216              LD R4,TXTEND;
08F2    4304 101C              SUB R4,TXTUNF;
08F6    9E08                   RET;
08F8
08F8                           %****************************************
08F8                           % DIVIDE SUBDE CHKSGN CKHLDE CHGSGN     *
08F8                           %****************************************
08F8
08F8                           DIVIDE: %HL/DE RES IN BC REMAINER IN HL
08F8    A149                   LD R9,R4;
08FA    8D88                   CLR R8;
08FC    9B58                   DIV RR8,R5;
08FE    A196                   LD R6,R9;
0900    A184                   LD R4,R8;
0902    9E08                   RET;
0904
0904
0904
0904                           CHKSGN:
0904    8444                   ORB RH4,RH4;
0906    9E0D                   RET PL;
0908                           CHGSGN:
0908    8544                   OR R4,R4;
090A    9E06                   RET ZR;
090C    A049                   LDB RL1,RH4;
090E    8D42                   NEG R4;
0910    8849                   XORB RL1,RH4;
0912    5E0D 0464              JP PL,QHOW;
0916    0806 8080              XORB RH6,80H;
091A    9E08                   RET;
091C
091C                           CKHLDE:
091C    A049                   LDB RL1,RH4;
091E    8859                   XORB RL1,RH5;
0920    ED01                   JR PL,CK1;
0922    AD54                   EX R4,R5;
0924                           CK1:
0924    8B54                   CP R4,R5;

BASSYS                         MACRO 8/8000 Z8000 ASSEMBLER   1.2.1E   PAGE 1
0926    9E08                   RET;
0928
0928                           %****************************
0928                           % SETVAL FIN ENDCHK ERROR  *
0928                           %****************************
0928
0928                           SETVAL:
```

Abb. 5.1-1

```
0928  D2AF            CALR TSTV;
092A  5E07 096A       JP CY,QWHAT;
092E  93F4            PUSH R15↑,R4;
0930  D293 3D3D 0944  TESTC '=',SV1;
0936  D0E1            CALR EXPR;
0938  A146            LD R6,R4;
093A  97F4            POP R4,R15↑;
093C  2E4E            LDB R4↑,RL6;
093E  A940            INC R4,1;
0940  2E46            LDB R4↑,RH6;
0942  9E08            RET;
0944
0944                  SV1:
0944  5E08 096A       JP QWHAT;
0948
0948                  FIN:
0948  D29F 3B3B 0954  TESTC ';',FI1;
094E  97F1            POP R1,R15↑;
0950  5E08 0542       JP RUNSML;
0954
0954                  FI1:
0954  D2A5 0D0D 0960  TESTC 0DH,FI2;
095A  97F1            POP R1,R15↑;
095C  5E08 0534       JP RUNNXL;
0960
0960                  FI2:
0960  9E08            RET;
0962
0962                  ENDCHK:
0962  D2D6            CALR IGNB;
0964  0A09 0D0D       CPB RL1,0DH;
0968  9E06            RET ZR;
096A                  QWHAT:
096A  93F5            PUSH R15↑,R5;
096C                  AWHAT:
096C  7605 047C       LD R5,↑WHAT;
0970                  ERROR:
0970  8299            SUBB RL1,RL1;
0972  DF9E            CALR PRTSTG;
0974  97F5            POP R5,R15↑;
0976  2059            LDB RL1,R5↑;
0978  93F1            PUSH R15↑,R1;
097A  8299            SUBB RL1,RL1;
097C  2E59            LDB R5↑,RL1;
097E  6104 1008       LD R4,CURRNT;
0982  93F4            PUSH R15↑,R4;
0984  2049            LDB RL1,R4↑;
0986  A940            INC R4,1;
0988  0449            ORB RL1,R4↑;
098A  97F5            POP R5,R15↑;
098C  5E06 048C       JP ZR,RSTART;
0990  2049            LDB RL1,R4↑;
0992  8499            ORB RL1,RL1;
0994  5E05 06EE       JP MI,INPERR;
0998  DF5C            CALR PRTLN;
099A  AB50            DEC R5,1;
099C  97F1            POP R1,R15↑;
099E  2E59            LDB R5↑,RL1;
09A0  C93F            LDB RL1,'?';
09A2  DEFE            CALR OUTC;
09A4  8299            SUBB RL1,RL1;
09A6  DFB8            CALR PRTSTG;
```

Abb. 5.1-1

```
BASSYS                                    MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 1
09A8    5E08 048C          JP RSTART;
09AC                       QSORRY:
09AC    93F5               PUSH R15↑,R5;
09AE                       ASORRY:
09AE    7605 0484          LD R5,↑SORRY;
09B2    5E08 0970          JP ERROR;
09B6
09B6                       %*******************************
09B6                       % GETLN  FNDLN  UPR            *
09B6                       %*******************************
09B6
09B6                       GETLN:
09B6    DF08               CALR OUTC;
09B8    6105 1218          LD R5,BUFFER;
09BC                       GL1:
09BC    DEF3               CALR CHKIO;
09BE    0A09 0101          CPB RL1,1;     %DEL ZEICHEN
09C2    E611               JR ZR,GL3;
09C4    DF0F               CALR OUTC;
09C6    0A09 0A0A          CPB RL1,0AH;
09CA    E6F8               JR ZR,GL1;
09CC    8499               ORB RL1,RL1;
09CE    E6F6               JR ZR,GL1;
09D0    0A09 1B1B          CPB RL1,1BH;    %ESC
09D4    E60F               JR ZR,GL4;
09D6    2E59               LDB R5↑,RL1;
09D8    A950               INC R5,1;
09DA    0A09 0D0D          CPB RL1,0DH;
09DE    9E06               RET ZR;
09E0    A0D9               LDB RL1,RL5;
09E2    DE3F               CALR CXBUFE;
09E4    EEEB               JR NZ,GL1;
09E6                       GL3:
09E6    A0D9               LDB RL1,RL5;
09E8    DE3E               CALR CXBUFA;
09EA    E604               JR ZR,GL4;
09EC    AB50               DEC R5,1;
09EE    C908               LDB RL1,8;
09F0    DF25               CALR OUTC;
09F2    E8E4               JR GL1;
09F4                       GL4:
09F4    DF28               CALR CRLF;
09F6    C90B               LDB RL1,0BH;
09F8    5E08 09B6          JP GETLN;
09FC
09FC                       FNDLN:
09FC    8444               ORB RH4,RH4;
09FE    5E05 0464          JP MI,QHOW;
0A02    7605 121C          LD R5,↑TXTBGN;
0A06                       FNDLP:
0A06                       FL1:
0A06    93F4               PUSH R15↑,R4;
0A08    6104 101C          LD R4,TXTUNF;
0A0C    AB40               DEC R4,1;
0A0E    8B54               CP R4,R5;
0A10    97F4               POP R4,R15↑;
0A12    9E07               RET CY;
0A14    2059               LDB RL1,R5↑;
0A16    82C9               SUBB RL1,RL4;
0A18    A096               LDB RH6,RL1;
0A1A    A950               INC R5,1;
```

Abb. 5.1-1

```
0A1C    2059                LDB RL1,R5↑;
0A1E    B649                SBCB RL1,RH4;
0A20    E704                JR CY,FL2;
0A22    AB50                DEC R5,1;
0A24    8469                ORB RL1,RH6;
0A26    9E08                RET;
0A28
0A28                        FNDNXT:

BASSYS                                MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 2
0A28    A950                INC R5,1;
0A2A                        FL2:
0A2A    A950                INC R5,1;
0A2C                        FNDSKP:
0A2C    2059                LDB RL1,R5↑;
0A2E    0A09 0D0D           CPB RL1,0DH;
0A32    EEFB                JR NZ,FL2;
0A34    A950                INC R5,1;
0A36    E8E7                JR FL1;
0A38
0A38                        %***********************************
0A38                        %PRTSTG QTSTG PRTNUM PRTLN    UPR    *
0A38                        %***********************************
0A38
0A38                        PRTSTG:
0A38    A096                LDB RH6,RL1;
0A3A                        PS1:
0A3A    2059                LDB RL1,R5↑;
0A3C    A950                INC R5,1;
0A3E    8A69                CPB RL1,RH6;
0A40    9E06                RET ZR;
0A42    DF4E                CALR OUTC;
0A44    0A09 0D0D           CPB RL1,0DH;
0A48    EEF8                JR NZ,PS1;
0A4A    9E08                RET;
0A4C
0A4C                        QTSTG:
0A4C    D321 2222 0A74      TESTC '"',QT3;
0A52    C922                LDB RL1,22H;
0A54                        QT1:
0A54    5F00 0A38           CALL PRTSTG;
0A58    0A09 0D0D           CPB RL1,0DH;
0A5C    97F4                POP R4,R15↑;
0A5E    5E06 0534           JP ZR,RUNNXL;
0A62                        QT2:
0A62    2140                LD R0,R4↑; %VAR RESTURN
0A64    A941                INC R4,2;
0A66    0700 F000           AND R0,0F000H;
0A6A    0B00 E000           CP R0,0E000H; %JR BEFEHL
0A6E    1E46                JP ZR,R4↑;
0A70    A941                INC R4,2; %SONST JP DANACH
0A72    1E48                JP R4↑;
0A74                        QT3:
0A74    D335 2727 0A7E      TESTC 27H,QT4;
0A7A    C927                LDB RL1,27H;
0A7C    E8EB                JR QT1;
0A7E                        QT4:
0A7E    D33A 5F5F 0A8E      TESTC '←',QT5;
0A84    C98D                LDB RL1,8DH;     %CR OHNE LF
0A86    5F00 0BA8           CALL OUTC;
0A8A    97F4                POP R4,R15↑;
0A8C    E8EA                JR QT2;
```

Abb. 5.1-1

```
0A8E                  QT5:
0A8E   9E08           RET;
0A90
0A90                  PRTNUM:
0A90   C600           LDB RH6,0;
0A92   5F00 0904      CALL CHKSGN;
0A96   ED02           JR PL,PN1;
0A98   C62D           LDB RH6,'-';
0A9A   AAE0           DECB RL6,1;
0A9C                  PN1:
0A9C   93F5           PUSH R15↑,R5;
0A9E   2105 000A      LD R5,0AH;
0AA2   93F5           PUSH R15↑,R5;
0AA4   AAE0           DECB RL6,1;
0AA6   93F6           PUSH R15↑,R6;
0AA8                  PN2:
0AA8   D0D9           CALR DIVIDE;
BASSYS                        MACRO 8/8000 Z8000 ASSEMBLER 1.2.1E   PAGE 2
0AAA   8566           OR R6,R6;
0AAC   E605           JR ZR,PN3;
0AAE   2DF4           EX R4,R15↑;
0AB0   AAC0           DECB RL4,1;
0AB2   93F4           PUSH R15↑,R4;
0AB4   A164           LD R4,R6;
0AB6   E8F8           JR PN2;
0AB8                  PN3:
0AB8   97F6           POP R6,R15↑;
0ABA                  PN4:
0ABA   AAE0           DECB RL6,1;
0ABC   84EE           ORB RL6,RL6;
0ABE   E503           JR MI,PN5;
0AC0   C920           LDB RL1,' ';
0AC2   DF8E           CALR OUTC;
0AC4   E8FA           JR PN4;
0AC6                  PN5:
0AC6   A069           LDB RL1,RH6;
0AC8   8466           ORB RH6,RH6;
0ACA   E601           JR ZR,SKIP2;
0ACC   DF93           CALR OUTC;
0ACE                  SKIP2:
0ACE   A0CD           LDB RL5,RL4;
0AD0                  PN6:
0AD0   A0D9           LDB RL1,RL5;
0AD2   0A0D 0A0A      CPB RL5,0AH;
0AD6   97F5           POP R5,R15↑;
0AD8   9E06           RET ZR;
0ADA   0009 3030      ADDB RL1,'0';
0ADE   DF9C           CALR OUTC;
0AE0   E8F7           JR PN6;
0AE2                  PRTLN:
0AE2   205C           LDB RL4,R5↑;
0AE4   A950           INC R5,1;
0AE6   2054           LDB RH4,R5↑;
0AE8   A950           INC R5,1;
0AEA   CE04           LDB RL6,4;
0AEC   D02F           CALR PRTNUM;
0AEE   C920           LDB RL1,' ';
0AF0   DFA5           CALR OUTC;
0AF2   8299           SUBB RL1,RL1;
0AF4   D05F           CALR PRTSTG;
0AF6   9E08           RET;
0AF8
```

Abb. 5.1-1

```
0AF8                    %************************
0AF8                    % MVUP MVDOWN POPA PUSHA*
0AF8                    %************************
0AF8
0AF8                    MVUP:
0AF8   8B54             CP R4,R5;
0AFA   9E06             RET ZR;
0AFC   BA51 0068        LDIB R6↑,R5↑,R0;
0B00   E8FB             JR MVUP;
0B02
0B02                    MVDOWN:
0B02   8B65             CP R5,R6;
0B04   9E06             RET ZR;
0B06   AB50             DEC R5,1;
0B08   AB40             DEC R4,1;
0B0A   2059             LDB RL1,R5↑;
0B0C   2E49             LDB R4↑,RL1;
0B0E   E8F9             JR MVDOWN;
0B10
0B10                    POPA:
0B10   97F6             POP R6,R15↑;
0B12   97F4             POP R4,R15↑;
0B14   6F04 1010        LD LOPVAR,R4;
0B18   8544             OR R4,R4;
0B1A   E609             JR ZR,PP1;
```

Abb. 5.1-1

```
BASSYS                        MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 2
0B1C   57F0 1012        POP LOPINC,R15↑;
0B20   57F0 1014        POP LOPLMT,R15↑;
0B24   57F0 1016        POP LOPLN,R15↑;
0B28   97F4             POP R4,R15↑;
0B2A   6F04 1018        LD LOPPT,R4;
0B2E                    PP1:
0B2E   93F6             PUSH R15↑,R6;
0B30   9E08             RET;
0B32
0B32                    PUSHA:
0B32   7604 1046        LD R4,↑STKLMT;
0B36   5F00 0908        CALL CHGSGN;
0B3A   97F6             POP R6,R15↑;
0B3C   A1F0             LD R0,R15;
0B3E   8104             ADD R4,R0;
0B40   5E0F 09AC        JP NC,QSORRY;
0B44   6104 1010        LD R4,LOPVAR;
0B48   8544             OR R4,R4;
0B4A   E60A             JR ZR,PU1;
0B4C   53F0 1018        PUSH R15↑,LOPPT;
0B50   53F0 1016        PUSH R15↑,LOPLN;
0B54   53F0 1014        PUSH R15↑,LOPLMT;
0B58   53F0 1012        PUSH R15↑,LOPINC;
0B5C   6104 1010        LD R4,LOPVAR;
0B60                    PU1:
0B60   93F4             PUSH R15↑,R4;
0B62   93F6             PUSH R15↑,R6;
0B64   9E08             RET;
0B66
0B66
0B66                    %*********************
0B66                    % OUTC CHKIO         *
0B66                    %*********************
0B66
0B66                    START:
```

```
0B66    760F 11D8          LD R15,↑STACK;
0B6A    C9FF               LDB RL1,0FFH;
0B6C                       INIT:
0B6C    6E09 1006          LDB OCSW,RL1;
0B70    DFE6               CALR CRLF;
0B72    8299               SUBB RL1,RL1;
0B74    7605 0BF4          LD R5,↑MSG1;
0B78    D0A1               CALR PRTSTG;
0B7A    7604 0B66          LD R4,↑START;
0B7E    6F04 101A          LD RANPNT,R4;
0B82    7604 121C          LD R4,↑TXTBGN;
0B86    6F04 101C          LD TXTUNF,R4;
0B8A    7604 1476          LD R4,↑TXTE;
0B8E    6F04 1216          LD TXTEND,R4;
0B92    7604 1478          LD R4,↑BUFA;
0B96    6F04 1218          LD BUFFER,R4;
0B9A    7604 14B8          LD R4,↑BUFE;
0B9E    6F04 121A          LD BUFEND,R4;
0BA2    5E08 048C          JP RSTART;
0BA6
0BA6                       CRLF:
0BA6    C90D               LDB RL1,0DH;
0BA8                       OUTC:
0BA8    93F6               PUSH R15↑,R6;
0BAA    93F1               PUSH R15↑,R1;
0BAC    6009 1006          LDB RL1,OCSW;
0BB0    8499               ORB RL1,RL1;
0BB2                       OC2:
0BB2    EE03               JR NZ,OC3;
0BB4    97F1               POP R1,R15↑;
0BB6    97F6               POP R6,R15↑;
0BB8    9E08               RET;
0BBA
0BBA                       OC3:

BASSYS                               MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 2
0BBA    97F1               POP R1,R15↑;
0BBC    93F1               PUSH R15↑,R1;
0BBE    A09E               LDB RL6,RL1;
0BC0                       LPT:
0BC0    0A0E 0D0D          CPB RL6,0DH;
0BC4    E605               JR ZR,LINEF;
0BC6                       H1:
0BC6    A0E9               LDB RL1,RL6;
0BC8    D40F               CALR CO11;
0BCA    97F1               POP R1,R15↑;
0BCC    97F6               POP R6,R15↑;
0BCE    9E08               RET;
0BD0                       LINEF:
0BD0    C90D               LDB RL1,0DH;
0BD2    D414               CALR CO11;
0BD4    CE0A               LDB RL6,0AH;
0BD6    E8F7               JR H1;
0BD8
0BD8                       CHKIO:
0BD8    D419               CALR CI11;
0BDA    0609 7F7F          ANDB RL1,7FH;
0BDE    0A09 0202          CPB RL1,2;
0BE2    EE03               JR NZ,CI1;
0BE4    4D00 1006          COM OCSW;
0BE8    E8F7               JR CHKIO;
0BEA
```

Abb. 5.1-1

```
0BEA                        CI1:
0BEA    0A09 0303           CPB RL1,3;
0BEE    9E0E                RET NZ;
0BF0    5E08 048C           JP RSTART;
0BF4
0BF4                        MSG1:
0BF4    5244 4B20 5A38      WORD: 'RDK Z8000 BASIC 1.0 ';
0BFA    3030 3020 4241
0C00    5349 4320 312E
0C06    3020
0C08    0D0A                WORD: 0D0AH;
0C0A
0C0A                        %***********************
0C0A                        % TABELLEN  DIRECT EXEC*
0C0A                        %***********************
0C0A                        %
0C0A                        % BYTE UND WORDS
0C0A
0C0A                        TAB1:
0C0A    4C49 5354 00        BYTE: 'LIST',0;
0C0F    055C                WORD: LIST1;
0C11    5255 4E00           BYTE: 'RUN',0;
0C15    052E                WORD: RUN;
0C17    4E45 5700           BYTE: 'NEW',0;
0C1B    051E                WORD: NEW;
0C1D    4259 4500           BYTE: 'BYE',0;
0C21    0DAA                WORD: BYE;
0C23    4D45 4D00           BYTE: 'MEM',0; %NEU END
0C27    0D76                WORD: END1;
0C29
0C29                        TAB2:
0C29    4E45 5854 00        BYTE: 'NEXT',0;
0C2E    0662                WORD: NEXT;
0C30    4C45 5400           BYTE: 'LET',0;
0C34    076A                WORD: LET;
0C36    4946 00             BYTE: 'IF',0;
0C39    06DC                WORD: IFF;
0C3B    474F 544F 00        BYTE: 'GOTO',0;
0C40    054C                WORD: GOTO;
0C42    474F 5355 4200      BYTE: 'GOSUB',0;
0C48    05B4                WORD: GOSUB;
0C4A    5245 5455 524E      BYTE: 'RETURN',0;
0C50    00

BASSYS                                MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E    PAGE 2
0C51    05D4                WORD: RETURN;
0C53    5245 4D00           BYTE: 'REM',0;
0C57    06D8                WORD: REM;
0C59    464F 5200           BYTE: 'FOR',0;
0C5D    05F0                WORD: FOR1;
0C5F    494E 5055 5400      BYTE: 'INPUT',0;
0C65    06F8                WORD: INPUT;
0C67    5052 494E 5400      BYTE: 'PRINT',0;
0C6D    056E                WORD: PRINT;
0C6F    5354 4F50           BYTE: 'STOP',0
0C73    0005 28             WORD: STOP;
0C76    4341 4C4C 00        BYTE: 'CALL',0;
0C7B    0DB0                WORD: CALL1;
0C7D    4F55 5443 00        BYTE: 'OUTC',0;
0C82    0EB8                WORD: OUTCHAR;
0C84    4F55 5400           BYTE: 'OUT',0;
0C88    0DC0                WORD: OUT1;
```

Abb. 5.1-1

```
0C8A    4F24 00             BYTE: 'O$',0;
0C8D    0DEE                WORD: OO;
0C8F    4924 00             BYTE: 'I$',0;
0C92    0DFE                WORD: II;
0C94    504F 4B45 00        BYTE: 'POKE',0;
0C99    0E40                WORD: POKE;
0C9B    5441 4200           BYTE: 'TAB',0;
0C9F    0DD6                WORD: TAB;
0CA1    4259 5445 00        BYTE: 'BYTE',0;
0CA6    0E58                WORD: BYTE1;
0CA8    574F 5244 00        BYTE: 'WORD',0;
0CAD    0E60                WORD: WORD1;
0CAF    00                  BYTE: 0;
0CB0    0762                WORD: DEFLT;
0CB2
0CB2                        TAB4:
0CB2    524E 4400           BYTE: 'RND',0;
0CB6    08A6                WORD: RND;
0CB8    4142 5300           BYTE: 'ABS',0;
0CBC    08E4                WORD: ABS1;
0CBE    5349 5A45 00        BYTE: 'SIZE',0;
0CC3    08EE                WORD: SIZE;
0CC5    5045 454B 00        BYTE: 'PEEK',0;
0CCA    0E38                WORD: PEEK;
0CCC    494E 4300           BYTE: 'INC',0;
0CD0    0EC2                WORD: INCHAR;
0CD2    4845 5800           BYTE: 'HEX',0;
0CD6    0ECA                WORD: HEX;
0CD8    494E 00             BYTE: 'IN',0;
0CDB    0DE6                WORD: IN1;
0CDD    2700                BYTE: 27H,0;
0CDF    0E94                WORD: QUOTE;
0CE1    544F 5000           BYTE: 'TOP',0;
0CE5    0EA8                WORD: TOP;
0CE7    4C45 4E00           BYTE: 'LEN',0;
0CEB    0EB0                WORD: LENGTH;
0CED    4353 5453 00        BYTE: 'CSTS',0;
0CF2    0E8A                WORD: CSTAT;
0CF4    00                  BYTE: 0;
0CF5    087E                WORD: XP40;
0CF7                        TAB5:
0CF7    544F 00             BYTE: 'TO',0;
0CFA    0602                WORD: FR1;
0CFC    00                  BYTE: 0;
0CFD    096A                WORD: QWHAT;
0CFF
0CFF                        TAB6:
0CFF    5354 4550 00        BYTE: 'STEP',0;
0D04    0610                WORD: FR2;
0D06    00                  BYTE: 0;
0D07    0614                WORD: FR3;
0D09

BASSYS                             MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 2
0D09                        TAB8:
0D09    3E3D 00             BYTE: '>=',0;
0D0C    0782                WORD: XP11;
0D0E    2300                BYTE: '#',0;
0D10    078A                WORD: XP12;
0D12    3E00                BYTE: '>',0;
0D14    0792                WORD: XP13;
0D16    3D00                BYTE: '=',0;
```

Abb. 5.1-1

Abb. 5.1-1

```
0D18    07A8         WORD: XP15;
0D1A    3C3D 00      BYTE: '<=',0;
0D1D    079C         WORD: XP14;
0D1F    3C00         BYTE: '<',0;
0D21    07B0         WORD: XP16;
0D23    00           BYTE: 0;
0D24    07B8         WORD: XP17;
0D26
0D26
0D26                 %****************
0D26                 % DIRECT MODUL  *
0D26                 %****************
0D26
0D26                 DIRECT:
0D26    7604 0C09    LD R4,↑TAB1-1;
0D2A                 EXEC:
0D2A                 EX0:
0D2A    D4BA         CALR IGNB;
0D2C    93F5         PUSH R15↑,R5;
0D2E                 EX1:
0D2E    2059         LDB RL1,R5↑;
0D30    A950         INC R5,1;
0D32    0A09 2E2E    CPB RL1,'.';
0D36    E60D         JR ZR,EX3;
0D38    A940         INC R4,1;
0D3A    0A49         CPB RL1,R4↑;
0D3C    E6F8         JR ZR,EX1;
0D3E    8C98         CLRB RL1;
0D40    AB50         DEC R5,1;
0D42    0A49         CPB RL1,R4↑;
0D44    E60A         JR ZR,EX5;
0D46                 EX2:
0D46    A940         INC R4,1;
0D48    0A49         CPB RL1,R4↑;
0D4A    EEFD         JR NZ,EX2;
0D4C    A941         INC R4,2;
0D4E    97F5         POP R5,R15↑;
0D50    E8EC         JR EX0;
0D52                 EX3:
0D52    8C98         CLRB RL1;
0D54                 EX4:
0D54    A940         INC R4,1;
0D56    0A49         CPB RL1,R4↑;
0D58    EEFD         JR NZ,EX4;
0D5A                 EX5:
0D5A    A940         INC R4,1;
0D5C    2040         LDB RH0,R4↑;
0D5E    A940         INC R4,1;
0D60    2048         LDB RL0,R4↑;
0D62    97F1         POP R1,R15↑;
0D64    1E08         JP R0↑;
0D66
0D66                 %******************
0D66                 %END EXEC         *
0D66                 %******************
0D66
0D66
0D66                 %EINZELNE ROUTINEN
0D66
0D66
0D66                 CXBUFE:
```

```
BASSYS                                   MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 2
0D66    6100 121A          LD R0,BUFEND;
0D6A    8A89               CPB RL1,RL0;
0D6C    9E08               RET;
0D6E
0D6E                       CXBUFA:
0D6E    6100 1218          LD R0,BUFFER;
0D72    8A89               CPB RL1,RL0;
0D74    9E08               RET;
0D76
0D76                       END1:
0D76    D301               CALR EXPR;
0D78    7605 1476          LD R5,↑TXTE;
0D7C    8B54               CP R4,R5;
0D7E    5E07 09AE          JP CY,ASORRY;
0D82    8444               ORB RH4,RH4;
0D84    5E05 09AE          JP MI,ASORRY;
0D88    2140               LD R0,R4↑;
0D8A    0D40               COM R4↑;
0D8C    8D00               COM R0;
0D8E    0B40               CP R0,R4↑;
0D90    5E0E 09AE          JP NZ,ASORRY;
0D94    6F04 121A          LD BUFEND,R4;
0D98    0304 0084          SUB R4,132;
0D9C    6F04 1218          LD BUFFER,R4;
0DA0    AB41               DEC R4,2;
0DA2    6F04 1216          LD TXTEND,R4;
0DA6    5E08 048C          JP RSTART;
0DAA
0DAA                       BYE:
0DAA    7FFF               SC 255;
0DAC    5E08 048C          JP RSTART;
0DB0
0DB0                       CALL1:
0DB0    D31E               CALR EXPR;
0DB2    93F5               PUSH R15↑,R5;
0DB4    7606 0DBC          LD R6,↑HERE;
0DB8    93F6               PUSH R15↑,R6;
0DBA    1E48               JP R4↑;
0DBC                       HERE:
0DBC    97F5               POP R5,R15↑;
0DBE    D4FE               CALR FINI;
0DC0
0DC0                       OUT1: %NUR IN SYS MODE
0DC0    D298               CALR PARN;
0DC2    93F4               PUSH R15↑,R4;
0DC4    D4DD 3D3D 0DD2     TESTC '=',RSV0;
0DCA    D32B               CALR EXPR;

0DCC    97F0               POP R0,R15↑;
0DCE    3F04               OUT R0↑,R4;
0DD0    D507               CALR FINI;
0DD2                       RSV0:
0DD2    5E08 096A          JP QWHAT;
0DD6
0DD6                       TAB:
0DD6    D2A3               CALR PARN;
0DD8                       A1:
0DD8    8544               OR R4,R4;
0DDA    EE01               JR NZ,SK4;
0DDC    D50D               CALR FINI;
0DDE                       SK4:
```

Abb. 5.1-1

```
0DDE    AB40                DEC R4,1;
0DE0    C920                LDB RL1,' ';
0DE2    D11E                CALR OUTC;
0DE4    E8F9                JR A1;
0DE6
0DE6                        IN1:
0DE6    D2AB                CALR PARN;
0DE8    A140                LD R0,R4;
0DEA    3D04                IN R4,R0↑;
```

BASSYS MACRO 8/8000 Z8000 ASSEMBLER 1.2.1E PAGE 2

```
0DEC    9E08                RET;
0DEE
0DEE                        OO:
0DEE    D33D                CALR EXPR;
0DF0    93F5                PUSH R15↑,R5;
0DF2    AD54                EX R4,R5;
0DF4    8D18                CLR R1;
0DF6    5F00 0A38           CALL PRTSTG;
0DFA    97F5                POP R5,R15↑;
0DFC    D51D                CALR FINI;
0DFE
0DFE                        II:
0DFE    D345                CALR EXPR;
0E00    93F5                PUSH R15↑,R5;
0E02    6105 101C           LD R5,TXTUNF;
0E06    8B54                CP R4,R5;
0E08    5E07 09AE           JP CY,ASORRY;
0E0C    6105 1218           LD R5,BUFFER;
0E10    D22B                CALR GL1;
0E12    A146                LD R6,R4;
0E14    AD54                EX R4,R5;
0E16    AB40                DEC R4,1;
0E18    6105 1218           LD R5,BUFFER;
0E1C    93F5                PUSH R15↑,R5;
0E1E    D194                CALR MVUP;
0E20    0C68                CLRB R6↑;
0E22    97F5                POP R5,R15↑;
0E24    A940                INC R4,1;
0E26    8354                SUB R4,R5;
0E28    AD54                EX R4,R5;
0E2A    7604 1000           LD R4,↑LEGT;
0E2E    2E4D                LDB R4↑,RL5;
0E30    A940                INC R4,1;
0E32    2E45                LDB R4↑,RH5;
0E34    97F5                POP R5,R15↑;
0E36    D53A                CALR FINI;
0E38
0E38                        PEEK:
0E38    D2D4                CALR PARN;
0E3A    204C                LDB RL4,R4↑;
0E3C    8C48                CLRB RH4;
0E3E    9E08                RET;
0E40
0E40                        POKE:
0E40    D366                CALR EXPR;
0E42    93F4                PUSH R15↑,R4;
0E44    D51D 2C2C 0E54      TESTC ',',PK1;
0E4A    D36B                CALR EXPR;
0E4C    A0C9                LDB RL1,RL4;
0E4E    97F4                POP R4,R15↑;
0E50    2E49                LDB R4↑,RL1;
```

Abb. 5.1-1

```
0E52    D548                CALR FINI;
0E54                        PK1:
0E54    5E08 096A           JP QWHAT;
0E58
0E58                        BYTE1:
0E58    D2E4                CALR PARN;
0E5A    A0C9                LDB RL1,RL4;
0E5C    DFF9                CALR WRIT2;
0E5E    D54E                CALR FINI;
0E60
0E60                        WORD1:
0E60    D2E8                CALR PARN;
0E62    A049                LDB RL1,RH4;
0E64    DFFD                CALR WRIT2;
0E66    A0C9                LDB RL1,RL4;
0E68    DFFF                CALR WRIT2;
0E6A    D554                CALR FINI;
0E6C

BASSYS                              MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 2
0E6C                        WRIT2:
0E6C    93F1                PUSH R15↑,R1;
0E6E    B311 FBFC           SRL R1,4;
0E72    DFFF                CALR IST;
0E74    97F1                POP R1,R15↑;
0E76                        IST:
0E76    0609 0F0F           ANDB RL1,0FH;
0E7A    0009 9090           ADDB RL1,90H;
0E7E    B090                DAB RL1;
0E80    C840                LDB RL0,40H;
0E82    B489                ADCB RL1,RL0;
0E84    B090                DAB RL1;
0E86    5E08 0BA8           JP OUTC;
0E8A
0E8A                        CSTAT:
0E8A    5F00 03B0           CALL CSTS;
0E8E    A09C                LDB RL4,RL1;
0E90    8C48                CLRB RH4;
0E92    9E08                RET;
0E94
0E94                        QUOTE:
0E94    2059                LDB RL1,R5↑;
0E96    A950                INC R5,1;
0E98    A09C                LDB RL4,RL1;
0E9A    8C48                CLRB RH4;
0E9C    D549 2727 0EA4      TESTC 27H,ASCI;
0EA2    9E08                RET;
0EA4                        ASCI:
0EA4    5E08 096A           JP QWHAT;
0EA8
0EA8                        TOP:
0EA8    6104 101C           LD R4,TXTUNF;
0EAC    A940                INC R4,1;
0EAE    9E08                RET;
0EB0
0EB0                        LENGTH:
0EB0    6104 1000           LD R4,LEGT;
0EB4    AB40                DEC R4,1;
0EB6    9E08                RET;
0EB8
0EB8                        OUTCHAR:
0EB8    D3A2                CALR EXPR;
```

Abb. 5.1-1

```
0EBA    A0C9                  LDB RL1,RL4;
0EBC    5F00 0BA8             CALL OUTC;
0EC0    D57F                  CALR FINI;
0EC2
0EC2                          INCHAR:
0EC2    D176                  CALR CHKIO;
0EC4    A09C                  LDB RL4,RL1;
0EC6    8C48                  CLRB RH4;
0EC8    9E08                  RET;
0ECA
0ECA                          HEX:
0ECA    93F6                  PUSH R15↑,R6;
0ECC    8D48                  CLR R4;
0ECE    D562 2828 0EF4        TESTC '(',HN2;
0ED4                          HNXTH:
0ED4    2059                  LDB RL1,R5↑;
0ED6    0A09 0D0D             CPB RL1,0DH;
0EDA    5E06 096A             JP ZR,QWHAT;
0EDE    DFF4                  CALR CNVBN;
0EE0    B349 0004             SLA R4,4;
0EE4    8C18                  CLRB RH1;
0EE6    8114                  ADD R4,R1;
0EE8    A950                  INC R5,1;
0EEA    D570 2929 0ED4        TESTC ')',HNXTH;
0EF0                          POPRET:
0EF0    97F6                  POP R6,R15↑;
0EF2    9E08                  RET;

BASSYS                                  MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 2
0EF4
0EF4                          HN2:
0EF4    5E08 096A             JP QWHAT;
0EF8
0EF8                          CNVBN:
0EF8    0A09 3030             CPB RL1,'0';
0EFC    5E05 096A             JP MI,QWHAT;
0F00    0A09 3939             CPB RL1,'9';
0F04    E509                  JR MI,CONTC;
0F06    E608                  JR ZR,CONTC;
0F08    0A09 4141             CPB RL1,'A';
0F0C    5E05 096A             JP MI,QWHAT;
0F10    0A09 4747             CPB RL1,'G';
0F14    5E0D 096A             JP PL,QWHAT;
0F18                          CONTC:
0F18    0209 3030             SUBB RL1,'0';
0F1C    0A09 0A0A             CPB RL1,10;
0F20    9E05                  RET MI;
0F22    0209 0707             SUBB RL1,7;
0F26    9E08                  RET;
0F28
0F28                          CONT:
0F28    5F00 03B0             CALL CSTS;
0F2C    8511                  OR R1,R1; %FLAGS SETZEN
0F2E    9E06                  RET ZR;
0F30    5F00 037C             CALL CI;
0F34    0A09 0303             CPB RL1,3;
0F38    9E0E                  RET NZ;
0F3A    5E08 048C             JP RSTART;
0F3E
0F3E
0F3E
0F3E
```

Abb. 5.1-1

```
0F3E            ORIGIN 1000H^;
1000
1000            % SPEICHERZELLEN
1000
1000            LEGT:    WORD (1);
1002            IOBUFA:  BYTE (1);
1003            IOBUFB:  BYTE (1);
1004            IOBUFC:  BYTE (1);
1005            LSTROM:  BYTE (1);
1006            OCSW:    BYTE (1);
1007            NOPNOP:  BYTE (1);
1008            CURRNT:  WORD (1);
100A            STKGOS:  WORD (1);
100C            VARNXT:  WORD (1);
100E            STKINP:  WORD (1);
1010            LOPVAR:  WORD (1);
1012            LOPINC:  WORD (1);
1014            LOPLMT:  WORD (1);
1016            LOPLN:   WORD(1);
1018            LOPPT:   WORD (1);
101A            RANPNT:  WORD (1);
101C            TXTUNF:  WORD (1);
101E            WORD (20);
1046            STKLMT:  WORD (1);
1048
1048            WORD (200);
11D8            STACK:   WORD (1);
11DA            VARBGN:  WORD (30);
1216            TXTEND:  WORD (1);
1218            BUFFER:  WORD (1);
121A            BUFEND:  WORD (1);
121C            TXTBGN:  WORD (1);
121E            WORD (300);
1476            TXTE:    WORD (1);
1478            BUFA:    BYTE (64);
14B8            BUFE:    BYTE (2);


BASSYS                    MACRO 8/8000 Z8000 ASSEMBLER  1.2.1E   PAGE 3
14BA            ORIGIN 1E00H^;
1E00
1E00            REGSTO:  WORD (16);
1E20            FCWSTO:  WORD (1);
1E22            PCSTO:   WORD (1);
1E24            INSTRW:  WORD (1);
1E26
1E26
1E26
1E26            END.
NEITHER WARNING NOR ERROR MESSAGES

****(EXECUTIVE)         NORMAL TERMINATION


A>^L
```

Abb. 5.1-1

5.1.1 Z8000-Monitor

Da es für den Z8000 noch keine Standard-Ein-Ausgabeprogramme gibt, wurde das Basic um einen Monitor erweitert, der je nach Hardware vom Leser leicht modifiziert werden kann. Der Monitor besitzt folgende Befehle:

Lexpr — Laden auf Adresse „expr". Ein Programm kann im Binärformat in den Speicher geladen werden. Damit kann zum Beispiel ein Programm von einem anderen Computer in den Z8000 übermittelt werden. Das Format lautet wie folgt: 00FF FF FF 00 LL LH d a t a.
„LL" bedeutet Anzahl der Datenbytes low order, „LH" sind die Anzahl der Datenbytes high order. „d a t a" sind die Daten, wobei immer erst die höherwertige Stelle (high order) und dann die niederwertige (low order) Stelle folgt.

Dexpr1 expr2 — Speicherzellen-Inhaltsanzeigen von Adresse „expr1" bis „expr2".

Eexpr — Eingabe von Maschinenprogrammen per Hand. Programme können „hex" (sedezimal), beginnend von Adresse „expr", eingegeben werden. Dafür gibt es noch einige Zusatzkommandos:

- CR — Beendet Eingabe ohne Abspeichern des angezeigten Wertes.
- , — Weiter (Adresse + 2) ohne Verändern
- – — Zurück (Adresse – 2) ohne Verändern
- space — Übergeben des Wertes in den Speicher. Wird keine Zahl eingegeben, so wird 0 abgespeichert. Der Speicherinhalt sowie die Adresse werden zur Kontrolle immer mit ausgegeben.

Gexpr — Starten eines Programmes bei der Adresse „expr". Dabei werden die aktuellen Register übergeben.

R — Es erfolgt die Ausgabe der aktuellen Register.

Der Z8000 besitzt eine phantastische Möglichkeit, um Standard-I/O-Funktionen anderen Programmen zur Verfügung zu stellen: den Supervisor-Call. Dies war bislang nur bei großen Computern der Fall. Das Basic benützt ausschließlich die Supervisor-Calls zum Aufruf von I/O-Routinen (außer im Falle des Direktportzugriffs). Folgende Supervisor-Calls werden zur Verfügung gestellt:

SYS 0 — Neustart des Monitorprogramms, kann als letzter Befehl in einem Anwenderprogramm verwendet werden.

SYS 1 — Eingabe eines Zeichens von der Konsole. Das Zeichen steht am Schluß im Register RL1.

SYS 2 — Ausgabe eines Zeichens. Das Zeichen steht zuvor im Register RL1.

SYS 3 — Konsolstatus. In R1 steht Null, falls kein Zeichen eingegeben wurde, sonst 8.

SYS 255 — Es werden alle Register auf der Konsole ausgegeben und das Monitorprogramm gestartet. Dieser Befehl kann zum Austesten von Programmen verwendet werden. Er ist wegen der günstigen Codierung (7FFFH) leicht zu merken.

5.1.2 Der Basic-Interpreter

Das Basic wird vom Monitorprogramm aus mit dem Befehl B gestartet. Wurde einmal ein Start durchgeführt, so kann z.B. nach einem Reset das Basic durch den Befehl N gestartet werden. Dabei wird ein bestehendes Basic-Programm nicht zerstört.

Der Basic-Interpreter beginnt auf der Adresse 03A4H. Dort befindet sich eine Sprungtabelle für die Konsolaufrufe, so daß diese leicht verändert werden können. Das Basic ist praktisch eine Eins-zu-eins-Übersetzung des 8080-Basic aus den vorhergehenden Kapiteln. Es wurden aber jene Funktionen vereinfacht, die sich mit dem Z8000 besser und schneller lösen lassen. Bei der Übersetzung wurde folgende Konvention fast immer eingehalten:

R1 ist der Akku,
R4 ist das Registerpaar HL,
R5 ist das Registerpaar DE,
R6 ist das Registerpaar BC.

Zur Funktionsbeschreibung kann die Erklärung des RDK-Basic verwendet werden, da auch die Markennamen beibehalten wurden. Das Z8000-Basic läßt sich ebenfalls in ein EPROM brennen und benötigt 2 KWorte RAM von Adresse 1000H an.

5.2 Literaturverzeichnis

[1] INTEL Datenhandbuch 1980/ 8080 Befehlssatz. Intel Semiconductor GmbH, Seidlstraße 27, 8000 München 2.

[2] Feichtinger Herwig: Basic für Mikrocomputer. Franzis-Verlag, München.

[3] Pelka Horst: Praxis mit Mikroprozessoren. 2. Auflage. Franzis-Verlag, München.

[4] Klein, Rolf-Dieter: Mikrocomputersysteme. 2. Auflage. Franzis-Verlag, München.

[5] Klein, Rolf-Dieter: Mikrocomputer-Hard- und Softwarepraxis. Franzis-Verlag, München.

[6] Klein, Rolf-Dieter: Einplatinen-Computer mit Z8000. ELEKTRONIK, Heft 16, S. 44...50, 1980.

[7] Klein, Rolf-Dieter: Längenbestimmung von Z80-Befehlen. ELEKTRONIK, Heft 23, S. 85...87, 1980.

[8] Klein, Rolf-Dieter: ELIZA – oder: Der Computer als Psychoanalytiker. Hobby-Computer Sonderheft 1 der ELO, FUNKSCHAU, ELEKTRONIK, S. 95, 96.

[9] ELIZA. Creative Computing, Juli/August 1977, S. 100 ff.

[10] The AM Z8000 Family Data Book. Am Z8001/2 Prozessor Instruction Set. Advanced Micro Devices, München.

[11] Dr. Dobb's Journal of COMPUTER Calistenics & Orthodontia. PCC, Box E, Menlo Park, CA 94025.

[12] Klein, Michael: Z80-Applikationsbuch. Franzis-Verlag, München.

[13] Pelka, Horst: Was ist ein Mikroprozessor. 4. Auflage. Pelka, Horst: Von der Schaltalgebra zum Mikroprozessor. 2. Auflage. Franzis-Verlag, München.

[14] Klein, Rolf-Dieter: Basic für 8080-Systeme. Basic # Basic: S-100-System mit schnellem Kassetten-Interface. Hobbycomputer Sonderheft 1, Franzis-Verlag, München.

5.3 Glossar

A

Access:
Zugriff, zum Beispiel zu einer bestimmten Speicherzelle.

A/D-Umsetzer:
Analog-Digital-Umsetzer. Eine analoge Größe wird in eine digitale umgewandelt, z.B. eine veränderliche Spannung in ein Datenwort.

Adresse:
Eine Bezeichnung für einen bestimmten Speicherplatz oder eines Speicherbereiches.

Akkumulator:
Es handelt sich um ein Register, über das arithmetische und meist auch logische Befehle ausgeführt werden können. Der Akkumulator hat direkte Verbindung mit dem Rechenwerk.

ALE:
Adress Latch Enable. Übernahme einer Adresse in einen Speicher durch dieses Signal.

ALGOL:
Algorithmic Language. Es handelt sich um eine Programmiersprache für technisch-wissenschaftliche Probleme. Sie ist auch auf Mikroprozessoren verfügbar, z.B. von Lifeboat für den 8080.

Alignment:
Abgleich, Justierung.

ALU:
Arithmetic Logic Unit, Rechenwerk. In diesem Teil des Rechners werden die arithmetischen und logischen Befehle ausgeführt.

APL:
Eine Programmiersprache für den technisch-wissenschaftlichen Bereich, speziell auch für den Dialogbetrieb.

APU:
Arithmetic Processor Unit. Ein Gleitkommarechner z.B.

ASCII:
Amercian Standard Code for Information Interchange. Ein häufig gebrauchter Code für Informationsübertragung. Auch mit ISO-7-Bit-Code bezeichnet (DIN 66003) [2].

Assembler:
Ein Übersetzungsprogramm, das eine maschinennahe Programmiersprache in den Maschinencode übersetzt.

Assoziativ-Speicher:
Ein Speicher, bei dem der Zugriff nicht über eine bestimmte Adresse, sondern über den Speicherinhalt erfolgt.

Asynchron:
Taktunabhängige Betriebsweise.

Available:
Verfügbar.

B

Bank selekt:
Kann der Speicher von einem Mikroprozessor mangels genügend Adressensignale nicht voll adressiert werden, so kann der Speicher in Bereiche (Banks) unterteilt werden, die dann durch einen eigenen Hardwaremechanismus angewählt werden.

BAS-Mischer:
Aus den Synchronsignalen und dem Videosignal wird ein genormtes Signal zur Ansteuerung von einem Videomonitor erzeugt.

Basic:
Beginners' All-purpose Symbolic Instruction Code. Eine einfache dialogorientierte Programmiersprache, die in großer Vielfalt für Mikrocomputer erhältlich ist.

Baud:
Messung des Datenflusses, wobei die Zeit zur Übertragung des kürzesten Elementes als Maß genommen wird. Führt jedes Element ein Bit, so ist die Baudrate zahlen-

mäßig der Anzahl Bits pro Sekunde, z.B. 1200 Bd bei manchen Kassetten-Interfaces. [4, 5]

Betriebssystem:
Darunter versteht man eine Reihe von Programmen, die es dem Computer ermöglichen, selbständig Programme zu bearbeiten. CP/M ist zum Beispiel ein einfaches Betriebssystem für den 8080/Z80.

Bildwiederholspeicher:
Bei einem Datensichtgerät ist dieses der Speicherteil, in dem die darzustellenden Zeichen stehen. Er muß sowohl vom CRT-Controller als auch von einer weiteren Steuereinheit (z.B. der CPU) ansprechbar sein.

Bit:
Binary Digit. Kleinste Informationseinheit.

Bootstrap-Loader:
Urlader. Ein Programm, das es dem Computer ermöglicht, Programme zu laden. Dabei wird meist zunächst ein weiterer Lader geladen, der dann seinerseits in der Lage ist, größere Programme zu laden. So wird zum Beispiel auch das CP/M-Betriebssystem geladen.

Branch:
Verzweigung, bedingter Sprungbefehl.

Buffer:
Puffer. Speicher, in dem Daten kurzzeitig festgehalten werden, oder auch Treiber zum Schalten von größeren Lasten, z.B. von Lampen oder Relais.

Bubble memory:
Magnetblasenspeicher.

Bus:
Sammelleitung, an die mehrere Bausteine angeschlossen werden können. Beispiele: Adressenbus, Datenbus, IEC-Bus.

Busy:
Besetzt, belegt, beschäftigt.

Byte:
8 Bits (= 2 Hex-Digits!).

C

Cartridge:
Kassette, z.B. Magnetband-Kassette, Speichermodul usw.

CCD memory:
Ladungsgekoppelte Speicher.

Clock:
Takt; Uhr.

COBOL:
Common Business Oriented Language. Eine Programmiersprache für vorwiegend kaufmännische Aufgaben. Sie ist unter dem Betriebssystem CP/M auch für Mikrocomputer erhältlich.

Compiler:
Ein Übersetzungsprogramm, das eine höhere Programmiersprache in den Maschinencode übersetzt, bevor das Programm gestartet wird.

Conditional:
Bedingt. Beispiel: Conditional Branch.

Controller:
Steuereinheit. Beim IEC-Bus ist es das Gerät, das auch Steuerinformation aussenden darf, um z.B. Daten von anderen Geräten anzufordern, also der Zentralrechner.

Conversion:
Übersetzung, Umcodierung.

Core:
(Magnet-) Kern (-Speicher).

CP/M:
Disk Operating System für den 8080,entwickelt von Digital Research.

CPU:
Central Processing Unit. Zentraleinheit eines Computers, bestehend aus Rechenwerk und Steuerwerk.

Cross-Assembler:
Ein Assembler, der nicht auf der Maschine, für die er den Maschinencode erzeugt, lauffähig ist. Es gibt so zum Beispiel einen Cross-Assembler für den 6800, der auf dem 8080 unter CP/M läuft.

Cross-Compiler:
Ein Compiler, der auf einer anderen Maschine läuft als auf der, für die er den Maschinencode erzeugt.

CRT:
Cathode Ray Tube. Bildröhre, auch Datensichtgerät.

Cursor:
Sichtmarke zur Kennzeichnung der aktuellen Schreibposition in Datensichtgeräten.

D

Daisy chain:
Kette. Verkettung, z.B.: Um Interrupt-Prioritäten festzulegen, können von einem Baustein zum anderen Signale geführt werden, die nicht nach dem Prinzip einer Busleitung arbeiten. Statt dessen wird durch geographische Anordnung die Priorität festgelegt.

Data acquisition:
Datenaufnahme, Datensammlung.

Data logger:
Meist ein Gerät zur Aufnahme von analogen Daten und deren Umwandlung in eine digitale Darstellung. Die Daten können dann von einem Rechner „abgeholt" werden.

Debugging:
Fehlersuche und -beseitigung („Entwanzen").

Dense graphic:
Mittlere Dichte bei Graphicdarstellungen, z.B. 256 x 128 Punkte.

Density:
Dichte. Zum Beispiel haben „double dense"-Floppies doppelte Datendichte.

Device:
Gerät, Einheit.

D/A-Umsetzer:
Digital-Analog-Umsetzer.

Dialoggerät:
Gerät zur direkten Datenein- und Ausgabe.

Digit:
Ziffer, Stelle; üblicherweise dargestellt durch 4 Bits.

Digitalisierer:
Ein Gerät zur Eingabe von graphischen Darstellungen, z.B. mit einem Griffel oder Fadenkreuz auf einer speziellen Digitalisierungsoberfläche.

Direct access:
Direkter Zugriff, z.B. auf den Bildwiederholspeicher durch die CPU.

Directory:
Inhaltsverzeichnis, z.B. einer Floppy Disk.

DMA:
Direct Memory Access. Direkter Zugriff auf den Speicher eines Rechners, wobei die Zugriffssteuerung nicht vom Rechner vorgenommen wird, sondern zum Beispiel von einer Peripherie-Einheit (z.B. Floppy-Laufwerk).

DOS:
Disk Operating System. Ein Programm, das es ermöglicht, mit einer Floppy oder einem Plattenspeicher zu arbeiten; CP/M(8080/Z80), FLEX(6800/6809), oder ISIS(8080) sind solche DOS.

Dreileiterhandshake:
Wird z.B. beim IEC-Bus verwendet, um einen Datenaustausch zu synchronisieren.

Drum storage:
Trommelspeicher (heute veraltet).

Dump:
Auszug eines Speicherinhalts, meist hexadezimal.

Durchsatz:
Anzahl der Operationen, die ein Computer in einer Zeiteinheit leistet.

Dynamische Speicher:
Bei solch einem Speicher muß die Information zyklisch aufgefrischt werden. Vorteil ist die Verfügbarkeit von hohen Speicherkapazitäten bei geringem Leistungsverbrauch und kleinem Preis im Vergleich zu statischen Speichern.

E

EAROM:
Electrically Alterable Read-Only Memory. Festwertspeicher, der sich nicht nur elektrisch programmieren, sondern auch elektrisch löschen läßt.

EBCDIC-Code:
Extended Binary Coded Decimal Interchange Code. Ein alphanumerischer 8-Bit-Code.

Editor:
Ein Programm zur Eingabe , Änderung und Ausgabe von Texten für Source-Programme oder Textverarbeitung.

ELIZA:
Ein Dialogprogramm, das einen Psychologen darstellt und zur Erprobung von künstlicher Intelligenz erstellt wurde.

Emulation:
Softwaremäßige Nachbildung eines Computers, so daß der Befehlssatz eines Computers, der nachgebildet wird, auf einem anderen verfügbar ist, wenn auch die Ausführungszeit im allgemeinen kleiner ist als auf dem realen Computer.

Enable:
Freigabe.

to enter:
Eingeben.

EOC:
End Of Conversion. Ein Signal, das das Ende einer Umwandlung anzeigt.

EPROM:
Erasable Programmable Read Only Memory. Ein mit ultraviolettem Licht löschbarer, aber elektrisch programmierbarer Festwertspeicher.

to erase:
löschen.

Error:
Fehler, Irrtum (Das von Computern am liebsten ausgegebene Wort).

Europakarte:
Leiterplatte mit genormten Format: 100 mm x 160 mm.

Evaluation module:
Entwicklungseinheit.

Even odd parity:
Gerade/ungerade Parität (binäre Quersumme).

Exorciser:
Hilfsgerät zur Entwicklung von Mikrocomputersystemen mit der CPU-Familie 6800.

Expression:
Ausdruck.

F

Fan-in:
Eingangslastfaktor.

Fan-out:
Ausgangslastfaktor. Er gibt an, wieviele Bausteine der gleichen Logikserie an einen Ausgang mit dem angegebenen Fan-out angeschlossen werden können.

Fifo:
First In/First Out-Speicher. Zuerst eingehende Daten werden auch zuerst wieder ausgegeben.

File:
Datei. Eine Ansammlung von Datengruppen, die in einer Datei abgelegt werden.

Firmware:
Software, die fest zur Funktionsfähigkeit eines Systems z.B. in einem ROM vorhanden ist.

Fixed-point:
Festkomma (z.B. auf Mark und Pfennige mit stets 2 Kommastellen).

Flag:
Eine Marke oder ein Flipflop zum „Festhalten" eines Zustands (Merker).

Floating-point:
Gleitkomma-Darstellung von Zahlen, meist mit Zehnerexponent.

Floppy Disk:
Ein billiger Massenspeicher mit Kapazitäten von 90 kBytes (Minifloppy) bis 2 MByte (große Floppy mit Quad dense).

Fortran:
Formula Translation. Eine problemorientierte Programmiersprache, die für technisch wissenschaftliche Aufgaben ausgelegt ist. Für Mikrocomputer gibt es eine Vielzahl von Fortran-Compilern.

Front Panel:
Bedienungsfeld.

G

Gate:
Verknüpfungsschaltung (NAND, NOR, EXOR usw.).

H

Handler:
Routine zur Kontrolle eines peripheren Gerätes.

Handshake:
Quittungsbetrieb. Methode, um Geräte mit verschiedenen Arbeitsgeschwindigkeiten durch den Austausch von Steuersignalen zu synchronisieren.

Hardcopy:
Kopie auf Papier. Zum Beispiel ein direkter Ausdruck des aktuellen Bildschirminhalts.

Hardware:
Damit sind alle Bauteile bzw. Geräte eines Systems gemeint.

hexadezimal:
Siehe sedezimal.

High order:
Höherwertige Stelle.

I

ICE:
In-Circuit-Emulator. Mit Hilfe einer „Nabelschnur", an deren Ende z.B. ein Sockel für die Z80-CPU ist, kann von einem Entwicklungssystem aus der Ablauf in einem Mikroprozessor-Anwendungssystem verfolgt werden.

IEC-Bus:
Schnittstellennorm, um 8 Bit parallel und byteseriell Daten austauschen zu können.

to include:
Beinhalten, einschließen.

Increment:
Schrittweises Erhöhen eines Wertes um 1.

Index-Register:
Ein Register zur Modifikation der Operandenadresse eines Befehls. Damit ergibt sich z.B. durch Addition des Inhalts des Index-Registers zum Adreßteil eines Befehls eine neue „effektive" Adresse.

Initialisierung:
Die Anfangsschritte in einem Programm, um definierte Startwerte zu erhalten, z.B. I/O-Ports, Stackpointer usw.

Input:
Eingabe.

Instruction:
Befehl, Anweisung.

Interface:
Schnittstelle. Mit Hilfe eines Interface können zwei Systeme (z.B. Computer und Drucker) einander angepaßt werden.

Interpreter:
Ein Interpreter ist ein Programm, das z.B. Befehle einer höheren Programmiersprache ausführt, ohne sie vorher in den Maschinencode zu übersetzen (vgl. Compiler).

Interrupt:
Unterbrechung. Durch einen Interrupt, den meist ein Peripheriegerät anfordert, wird das gerade laufende Programm un-

terbrochen und eine spezielle Unterbrechungsroutine ausgeführt. Danach erfolgt ein Rücksprung in das unterbrochene Programm.

J

Job:
Auftrag.

Joystick:
Ein Kreuzknüppel-Potentiometer, das zur Eingabe von Daten verwendet werden kann, vorwiegend in Kombination mit grafischen Geräten.

Jump:
Sprung.

Jumper:
Leitungsbrücke. Mit Jumper wird im Jargon eine Brücke zur Einstellung von Parametern, wie z.B. einer Perpherieadresse, bezeichnet.

K

Keyboard:
Tastatur.

Kit:
Bausatz.

Kompatibel:
austauschbar, aneinander angepaßt.

L

Label:
Marke. In Programmiersprachen ist damit meist eine symbolische Adresse (d.h. ein Name) gemeint, auf Magnetbändern z.B. ist damit ein Identitätskennzeichen gemeint.

Lichtgriffel:
Ein Stift, der zur Eingabe von Daten direkt über die Bildfläche geeignet ist. Dazu liefert er immer dann einen Puls, wenn der Schreibstrahl des Sichtgerätes auf die Optik des Griffels trifft.

Lifo:
Last In/First Out. Zuletzt gespeicherte Daten werden zuerst ausgegeben (Stack).

Linker:
Ein Programm, das mehrere Teilprogramme, die schon assembliert wurden, zu einem binden kann. Dazu muß der Assembler die nötige Information dem Linker übergeben können.

LISP:
List Processing. Eine Programmiersprache für die Verarbeitung von Listen und rekursive Datenstrukturen. Die Sprache eignet sich besonders für AI (artificial intelligence – künstliche Intelligenz). Das bekannte Programm ELIZA wurde z.B. zunächst in LISP programmiert.

Listener:
Name, der beim IEC-Bus verwendet wird. Ein Gerät, das Daten vom IEC-Bus empfangen kann, heißt Listener.

Listing:
Ausdruck. Auflistung.

Loader:
Ein Ladeprogramm.

Logic analyzer:
Ein Hilfsgerät zum Testen von Digitalschaltungen, mit einer Anzeige für die logischen Zustände in dieser Schaltung.

Loop:
Schleife. Durch einen Sprung rückwärts kann zum Beispiel eine Schleife in einem Programm entstehen.

Low order:
Niederwertige Stelle.

M

Makro:
Mit Hilfe einer Makroanweisung kann der Programmierer in Assemblern eine Folge von Befehlen definieren, die durch Angabe des Makronamens in dem Assemblerprogramm durch den Assembler bei der Über-

setzung eingefügt werden. Dabei können durch Parameterangaben die eingefügten Befehlssequenzen variiert werden.

Maschinencode:
Maschinensprache. Damit ist ein Binär-Code gemeint, der vom Mikrocomputer z.B. direkt verstanden wird.

Maske:
Ein Bitmuster, mit dem bestimmte Bitgruppen ausgeblendet, komplementiert oder eingefügt werden.

Memory:
Speicher.

Mikroprogrammierbar:
Der Befehlssatz mancher Prozessoren kann mit Hilfe von Mikrobefehlen definiert werden.

Mikrocomputer:
Besteht aus einem Mikroprozessor, Speichern und Peripherie.

Mikroprozessor:
Ein integrierter Baustein, als Teile eines Mikrocomputers, der ein Leit- und ein Rechenwerk besitzt. Der interne Ablauf kann in der Regel von außen durch Software beeinflußt werden.

Mnemonischer Code:
Leicht zu merkende Kurzwörter, deren Inhalt auf die Verwendung schließen läßt. Derartige Kurzwörter werden in Assemblersprachen eingesetzt, z.B. LDA, STA, JMP usw.

Modem:
Modulator und Demodulator. Eine Schaltung, die Daten für die Fernübertragung aufbereitet.

Monitorprogramm:
Ein Programm, das dem Benutzer den elementaren Umgang mit dem Computer ermöglicht.

Multiplex:
Übertragung von mehreren verschiedenen Informationen, die dazu zeitlich hintereinander übertragen werden.

Multiprozessing:
Aufbau eines Rechners mit mehreren CPUs oder Teilcomputern.

N

Nesting:
Verschachtelung; zum Beispiel Verschachteln von Interrupts oder Unterprogrammen.

O

Off-line:
Der Benutzer ist dabei nicht hardwaremäßig mit dem Computer verbunden, sondern der Verkehr wird über Datenträger abgewickelt.

Oktal:
Zahlendarstellung zur Basis 8.

On-line:
Dabei ist das Terminal des Benutzers über eine Datenleitung direkt mit dem Computer verbunden.

Output:
Ausgabe.

P

Packen:
Dabei werden zum Beispiel zwei Dezimalzahlen in einem Byte untergebracht, also „gepackt".

Parity:
Parität, Gleichheit.

Pascal:
Eine höhere Programmiersprache, die zunehmend Verbreitung auch bei Mikrocomputern findet.

Pass:
Lauf, z.B. eines Programms.

Peripherie:
Externe Datenend- und Speichergeräte.

PIA:
Peripheral Interface Adapter. Ein Baustein, der den Ein- und Ausgabeverkehr zwischen dem Mikroprozessor und der Peripherie abwickelt.

Pipelining:
Fließbandverarbeitung. Durch diese Verarbeitungsform kann die Ausführungszeit stark verkürzt werden. Während ein Befehl gerade ausgeführt wird, wird der nächste Befehl schon geholt. Bei Sprungbefehlen ergeben sich im allgemeinen allerdings zusätzliche Verzögerungen.

PL/1:
Programming Language. Eine höhere Programmiersprache, die ebenfalls für Mikrocomputer erhältlich ist.

PL/M:
Programming Language for Microcomputers. Eine höhere Programmiersprache, die auf der Basis von PL/1 arbeitet und speziell für Mikrocomputer entwickelt wurde.

Pointer:
Zeiger. Ein Speicherplatz, der eine Adresse enthält. Mit einem Zeiger lassen sich leicht Stacks aufbauen (Stack-Pointer).

Polling:
Aufrufbetrieb, Aufruftechnik. Um z.B. die Quelle eines Interrupts festzustellen, werden alle in Frage kommenden Quellen nacheinander abgefragt. Dieser Vorgang wird mit „polling" bezeichnet.

Port:
Tor. Schaltkreise für die Ein-/Ausgabe von Daten.

Power-on-Jump:
Nach Einschalten der Stromversorgung wird ein Sprungbefehl durch eine Hardwareschaltung vorgenommen, um z.B. in ein Monitorprogramm zu gelangen, dessen Anfangsadresse nicht mit der Adresse übereinstimmt, die nach einem Reset vom Prozessor angewählt wird.

Programm:
Eine Folge von Anweisungen (Befehlen), die zur Lösung eines Problems dienen sollen.

Programmierbarer Zeichengenerator:
Der Zeichensatz kann durch den Prozessor frei programmiert werden. Damit ist es einfach möglich, Sonderzeichen, Schaltsymbole im Programm selbst festzulegen und zur Darstellung zu gebrauchen.

Programmiersprachen:
Eine Sprache zur Formulierung von Programmen, die automatisch in die Maschinensprache übersetzt werden können.

PROM:
Programmable Read Only Memory. Ein programmierbarer Festwertspeicher.

Pseudobefehl:
Eine Instruktion, die eigentlich gar nicht vorhanden sein dürfte (gemäß den CPU-Herstellerangaben).

to punch:
Stanzen, lochen.

Q

Queue:
Warteschlange. Daten werden in einer Warteschlange angesammelt, solange sie noch nicht verarbeitet sind.

R

RALU:
Register und Arithmetic Logic Unit. Ein Prozessorelement mit einer ALU und einigen Registern.

RAM:
Random Access Memory. Ein Schreib-/Lesespeicher mit wahlfreiem Zugriff.

Reader:
(Lochstreifen oder Lochkarten) Leser.

Real Time:
Echtzeit. Arbeitsweise eines Computers.

Real time clock:
Echtzeituhr; eine Uhr, die die tatsächliche Uhr bereitstellt.

Redundanz:
Teil einer Nachricht, die zur eigentlichen Information nichts mehr beiträgt. Sie kann aber zum Beispiel zur Fehlererkennung oder Korrektur verwendet werden.

Refresh:
Wiederauffrischung. Wird bei dynamischen Speichern benötigt, um einen Informationsverlust zu verhindern.

Relokalisierbar:
Ein Programm, das auf unterschiedlichen Speicherplätzen direkt lauffähig ist, heißt relokalisierbar.

REPROM:
Reprogrammable Read Only Memory. Ein Festwertspeicher, der sich löschen und wieder neu programmieren läßt.

Request:
Anfordern. Anforderung.

to reset:
Rücksetzen, in den Grundzustand bringen.

Resident:
Im eigenen System fest vorhanden, z.B. bei einem residenten Assembler.

ROM:
Read Only Memory. Ein Festwertspeicher, von dem nur gelesen werden kann.

Run:
Durchlauf.

S

to scan:
Abtasten.

Schnittstelle:
Pegel- und anschlußgenormte Verbindungsstelle zwischen zwei Geräten.

sedezimal:
Zahlendarstellung zur Basis 16 mit den Ziffern 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F; häufig auch mit Hexadezimalsystem bezeichnet.

to select:
Auswählen.

to sense:
Abtasten.

Simulator:
Ein Programm, das einen Befehlssatz simuliert, den die CPU eigentlich nicht beherrscht.

Software:
Hierunter versteht man alle Arten von Programmen, wie auch Texte und Information.

Source:
Quelle.

Space:
Freiraum.

Sparse graphic:
Eine niedrig auflösende Grafik, z.B. 128 x 128 Punkte.

Stack:
Stapelspeicher, Kellerspeicher. Merkmal für einen Stack ist, daß eine Informationseinheit immer nur an der Stelle entnommen werden kann, an der sie zuletzt hinzugefügt wurde (LIFO).

State:
Zustand. Operationsschritt.

Statement:
Anweisung, Befehl.

Statischer Speicher:
Ein Speicherbaustein, der keine Wiederauffrischzyklen benötigt.

Steuerwerk:
Dieser Teil im Computer kontrolliert die Ausführung sämtlicher Befehle. Er wird auch mit Leitwerk bezeichnet.

Subroutine:
Unterprogramm.

Super dense graphic:
Hoch auflösende Grafik, z.B. 256 x 256 Bildpunkte.

Supervisor:
Ein Organisationsprogramm.

Synchron:
Ein Takt steuert den genauen Ablauf.

Syntaxdiagramme:
Eine besondere Art in Diagrammen Sprachbeschreibungen durchzuführen. Wurde durch die Pascal-Syntaxdiagramme bekannt.

T

Talker:
Name aus der IEC-Bus-Definition; ein Gerät, das Daten auf diesen Bus senden kann, heißt Talker.

Tape:
Ein Magnetband oder Lochstreifen.

Terminal:
Datenendstation. Ein Gerät zur Dateneinund/oder Datenausgabe.

Text-Editor:
siehe Editor.

Time sharing:
Zeitscheibenverfahren. Dabei können mehrere Benutzer auf ein und denselben Computer zugreifen.

Timing:
Zeitablauf.

Tiny:
Klein. Tiny Basic zum Beispiel bedeutet eine Teilmenge des Standard-Basic.

Trace:
Ablaufverfolgung. Ein Programm kann durch die schrittweise Ausführung und Protokollierung überwacht und damit ein Fehler leichter gefunden werden.

Track:
Spur, Bahn; eine Floppy ist beim 8-inch-Format in 76 Tracks unterteilt.

to transfer:
Übertragen.

U

UART:
Universal Asynchronous Receiver / Transmitter. Diese Schaltung übernimmt die Serien/Parallel- sowie die Parallel /Serienwandlung für eine asynchrone Datenübertragung.

Unit:
Gerät, Einheit.

Unterprogramm:
Gleiche Befehlsfolgen, die in einem Programm mehrmals vorkommen, werden im allgemeinen als Unterprogramm ausgeführt, so daß diese nur einmal im Programm auftreten und mit einem Unterprogrammsprung vom Hauptprogramm aufgerufen werden.

V

V24:
Schnittstellen-Norm für seriellen Datenaustausch.

Valid:
Gültig.

Vektor-Interrupt:
Von dem Gerät, das den Interrupt an den Prozessor gibt, wird zusätzlich ein Interruptvektor mitgeliefert, der dem Prozessor sagt, welche Unterbrechungsroutine ausgeführt werden soll.

Volatile:
Flüchtig, z.B. Halbleiterspeicher beim Abschalten der Versorgungsspannung.

W

Winchester Drive:
Ein spezielles Verfahren, bei dem es durch hermetischen Abschluß gelingt, hohe Speicherdichten auf kleinem Platz zu er-

reichen. Z.B. 8 inch Platte von IMI mit 22 M Bytes Kapazität auf drei Oberflächen.

Worst case:
Ungünstigster Fall.

Wort:
Zusammenfassung mehrerer Bits, sie können meist auch zusammenfassend verarbeitet werden (Z8000 CPU z.B. hat eine Wortbreite von 16 Bit).

X

XY-Recorder:
Ein XY-Schreiber bzw. ein Plotter, der Kurven auf Papier ausgeben kann.

Z

Zeichengenerator:
Meist ein Festwertspeicher, in dem ein Zeichensatz binär z.B. in Matrixform gespeichert ist.

Zugriff:
Zugang z.B. zu einer bestimmten Speicherzelle.

Zyklus:
Eine Anzahl von Schritten, die wiederholt werden und im Ablauf gewisse Änlichkeiten aufweisen.

Direkt-Befehle:

LIST	Listen eines Basic-Programms
RUN	Ausführen eines Basic-Programms
NEW	Löschen des Basic-Programms
BYE	Beenden des Basics
END	Speicherplatz neu besetzen

Programmierbare Befehle:

NEXT	Schleifenende
LET	Anfang einer arithmetischen Zuweisung
IF	Bedingung
GOTO	Unbedingter Sprung
GOSUP	Unterprogrammsprung
RETURN	Unterprogrammrücksprung
REM	Kommentar
FOR	Schleifenanfang
INPUIT	Eingabe
PRINT	Ausgabe
STOP	Beenden des Ablaufs
CALL	Unterprogrammaufruf – Maschinenprogramm
OUTCHAR	Einzelzeichenausgabe
OUT	Ausgabe an Port
O$	Ausgabe eines Textes
I$	Eingabe eines Textes
POKE	Direkter Speicherzugriff
TAB	Tabulator ausgeben
BYTE	Sedezimalausgabe in Byteform
WORD	Sedezimalausgabe in Wortform

Funktionen:

RND	Zufallswert
ABS	Absolutbetrag
SIZE	Speicherfreiraum
PEEK	Direktspeicherzugriff
INCHAR	Einzelzeicheneingabe
HEX	Umwandlung vom Sedezimalsystem
IN	Porteingabe
‘ ’	Einzelzeichenumwandlung
TOP	Erste freie Speicherzelle
LEN	Länge eines Strings
CSTS	Konsolstatus

Divers:

TO	In Schleifenanweisung
STEP	In Schleifenanweisung

Vergleichsbefehle:

>=
#
>
=
<=
<

Sachverzeichnis

A

ABS 70, 75
ADD 10
Anpassung 115
Arithmetische Ausdrücke 70
Ausdruck 74
Ausdrücke
  arithmetisch 70
Ausgabe 85, 86

B

BASIC 9, 10
– 12K 103
– 12K; Beispiel 125
– 12K; Operatoren 124
– 12K Befehlsbeschreibung 116
– Anpassung 115
– Realisierung 10
– Standard 31
– Tiny 31
– Z8000 129
BCD FLOATING POINT
  PACKAGE 94
Befehle einzeln 78
Befehlsabarbeitung 77
BYE 81
BYTE 88

C

CALL 86
CBM 29
CHKIO 67
CI 13, 67
CO 13, 67
Compiler 19
– Ausgabebeispiel 28
– Mini 20
CRLF 11
CSTAT 77

D

Datei 101
Define 26
Division 100
DIRECT 78
DOUBLE PRECISION 70

E

ECHO 67
Editor 117
Eingabe 84
ELIZA 125
END 81
EXPR 72, 75

F

FAC(x) 71
Fakultät 71
FIN 79
FINI 79
Fließenkommaarithmetik 94
Floppy-Anschluß 101
FNDLN 78
FNDNXT 78
FOR 83
Funktion 73, 75

G

GETHEX 11
GETLN 69, 70
Gleitkommaarithmetik 94
– Rechenbeispiele 99
Gleitkommadarstellung 97
GOSUB 82
GOTO 82
Graphik 101

H

HALT 11, 27
HEX 76

I

I$ 87
IF 82
IN 76
INCHAR 76
INPUT 84
INTEGER 70
Insert 27
Interpreter 10
I/0 Routinen 66

J

JMPNZ 10
JUMP 10

L

LENGTH 77
LET 78, 79
Linker 29
LIST 80
LOAD 10, 101

M

Makro 19, 26
Maschinenunterprogramm 86
Mini Compiler 20
Mini Interpreter 14
Monitor
– Z8000 163
– ZAPPLE 66
– programm 32
Multiplikation 68, 100
MVDOWN 78
MVUP 78

N

NEG 10
NEW 81
NEXT 83

O

OCSW 67
O$ 87
OUT 86
OUTC 67
OUTCHAR 86
OUTHEX 11

P

PARN 68
PEEK 76
PET 29
POKE 88
Port
  8080; Z80 76
PR 85
PRINT 85
– USING 118
PRTNUM 68, 69, 70
PRTSTG 70
Pointer 78

Q

QHOW 75
QTSTG 85
QUOTE 76
QWHAT 75

R

RDK-BASIC
  Befehlsübersicht 80
RDK-BASIC 31
  Beispiele 89
REAL 70
Rekursion 71
REM 83
RETURN 82
RND 75
RSTART 67, 77
RUN 10, 81
RUNNXL 79
RUNSML 79
RUNTSL 82

S

SAVE 101
SETVAL 78
Schleife 83
SIZE 68, 76
SPC 88
Sprache
  hypothetisch 19
Sprung 82
Standard Basic 31
START 67, 77
STARTE 13
STOP 86
STORE 10
String 84, 87
Syntaxdiagramm 71

T

TAB 88
TDL 103
Teilübersetzung 29
Tiny Basic 31
TOP 76, 87
TSTC 67
TSTNUM 68
TSTV 67

U

Unterprogramm 82

W

WORD 88

X

XP 75

Z

Z80 31
Z8000 31
– Basic 129
– Monitor 163
ZAPPLE Monitor 66
ZEICH 11
Zeichen|ausgabe 86
– verarbeitung 67
Zeilennummern 30

Weitere Franzis Elektronik-Fachbücher

Von der Schaltalgebra zum Mikroprozessor

Die Mikroprozessoren und ihre festverdrahtete und programmierbare Logik.
Von **Horst Pelka**
2., verbesserte und erweiterte Auflage. 339 Seiten mit 178 Abbildungen und 24 Tabellen.
Lwstr-kart. DM 28,–
ISBN 3-7723-6422-5

Mathematische Logik und elektronische Technik ergeben einen Mikroprozessor. Hier sind die Grundlagen dazu umfassend und doch kompakt dargestellt. Ausgegangen wird von den binären Zahlensystemen und Codes, um so in die Grundlagen der Digitaltechnik einzudringen. Auf die verschiedenen bipolaren und MOS-Technologien integrierter Schaltungen wird ebenso eingegangen, wie auf die Schaltungstechnische Realisierung von Verknüpfungsgliedern. Flip-Flops, Schieberegister und Zeitschaltungen. Fast die Hälfte des Buches behandelt die Grundlagen und Programmierung von Mikroprozessoren. Der Stoff ist einfach und klar dargestellt, viele Programmbeispiele erleichtern das Verständnis.
Mit diesem Buch lernt der Leser das Gebiet der festverdrahteten Logik und das der Mikroprozessoren kennen. Es ist ihm möglich, Entscheidungen bei der Auswahl dieser festverdrahteten und programmierbaren Logik zu treffen.

ABC der Mikroprozessoren und Mikrocomputer

Neue Fachwörter und Abkürzungen für Elektroniker, Programmierer und Praktiker verständlich gemacht.
Von **Horst Pelka**
(= RPB electronic-taschenbuch Nr. 135)
159 Seiten mit 45 Abbildungen.
Kart. DM 10,80
ISBN 3-7723-1351-5

Wer mit Mikroprozessoren und Mikrocomputern zu tun hat, sollte dieses Taschenbuch immer bei sich tragen. Dann sind Amerikanismen und Buchstabenkürzel jederzeit verständlich.

Während einer Diskussion ist der Band ein hervorragender Spickzettel, der brillieren läßt. Bei der Lektüre von Fachzeitschriften, Firmendruckschriften und Bedienungsanleitungen ist er ein zuverlässiges Dictionary und auch Glossarium. Die Zahl der Stichworte ist sehr groß.

Was ist ein Mikroprozessor?

Über die Arbeitsweise, Programmierung und Anwendung von Mikrocomputern.
Von **Horst Pelka**
(= RPB electronic-taschenbuch Nr. 82)

5., neubearbeitete und erweiterte Auflage. 130 Seiten mit 58 Abbildungen und 5 Tabellen.
Kart. DM 8,80
ISBN 3-7723-0825-2

Ja, was ist nun ein Mikroprozessor? Seit wann gibt es ihn überhaupt? Wie ist er entstanden? Hat ihn jemand erfunden? Wie ist er aufgebaut? Welche Technologien werden dabei verwendet? Können einzelne Bausteine ausgetauscht werden? Ist er eine Fortentwicklung der Mikrocomputersysteme? Was ist ein 1-Chip-Mikrocomputer? Wie ist seine Arbeitsweise? Ist er programmierbar? Wie weit? Welche Programmiersprachen versteht er? Können seine Programme geändert werden? Kann das jedermann? Ist die Programmierung stromausfallsicher?

Was leistet ein Mikroprozessor? Wie und wo kann er angewandt werden? Wird er den Taschencomputer ersetzen? Gibt es bereits Standardtypen? Inwieweit unterscheiden sie sich voneinander? Welches sind die Auswahlkriterien für Anwendung?
Fragen über Fragen! Die Antworten gibt dieser Band.

Franzis-Verlag München

Klein
Basic-Interpreter


Rolf-Dieter Klein

Wie man 8080- oder Z-80-Systeme nachträglich mit einem Basic-Interpreter ausrüsten kann, beschreibt dieses Buch. Dabei werden mehrere Ausführungen erörtert und beschrieben. Die beiden interessantesten sind: Ein Tiny-Basic-Interpreter und ein komfortabler 12-KByte-Basic-Interpreter. Der letztere ist einer der leistungsfähigsten und ist deshalb im Objektformat aufgelistet wiedergegeben. Die Krönung bildet die ausführliche Beschreibung eines Basic-Interpreters für den 16-Bit-Prozessor Z8000.
Großen Wert hat der Autor auf die Implementierung der unterschiedlichen Systemkonfigurationen gelegt. Damit wird der Leser dieses Buches in die Lage versetzt, auch mal selber einen Interpreter für seine Zwecke zu konstruieren. Er kann auch besser mit Assembler programmieren, da dazu sehr viele Programmiertricks gezeigt werden.

ISBN 3-7723-6941-3

Franzis'